The Dissertation Committee for Yung Yi
certifies that this is the approved version of the following dissertation:

Fluid Models for Internet Design and Simulation

Committee:

Sanjay Shakkottai, Supervisor

Ari Arapostathis

James C. Browne

John Hasenbein

Gustavo de Veciana

Sriram Viswanath

Fluid Models for Internet Design and Simulation

by

Yung Yi, B.S.E, M.S.E

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2006

To my wife Joohyun and my son David

Acknowledgments

I would like to express my sincere gratitude and appreciation to my advisor, Prof. Sanjay Shakkottai, for his support, patience, and encouragement throughput my graduate studies. I would not finish my Ph.D without his expert guidance and generous support. I am extremely proud and honored to have the opportunity to associate my name with him. I can also never thank Prof. Gustavo de Veciana enough. It is a great privilege and joy to work with Prof. Gustavo de Veciana. His dedication to research has inspired me to pursue ideas and research.

I thank my thesis committee members, Prof. Ari Arapostathis, Prof. James C. Browne, Prof. John Hasenbein, and Prof. Sriram Viswanath. They were very kind to agree to be on my committee, and I am grateful to them for their time and effort at various stages of my dissertation. I specially thank my M.S adviser, Prof. Yanghee Choi at Seoul National University for his valuable advice and support.

I appreciate the friendship from my colleagues at WNCG group members I have come to know during the past years. I want especially to thank our group members (Sundar Subramanian, Chang Woo Yang, Sandeep Bhadra, Shreeshankar Bodas, and Jung Ho Ryu). I also thank Chulhan Lee for his kindness to discuss anything related to my research.

I am deeply indebted to my dear wife Joohyun for her love, support, and encouragement. Last, but not the least, I thank my son David for the joy and the happiness he brings to me.

Yung Yi

The University of Texas at Austin

August 2006

Fluid Models for Internet Design and Simulation

Publication No. _____

Yung Yi, Ph.D.
The University of Texas at Austin, 2006

Supervisor: Sanjay Shakkottai

Over the past decade, fluid models have been widely used, and shown to be efficient and accurate in the modeling, analysis, and design of the Internet. In literature, much of this work has focused on the design of end-host controllers and control algorithms at routers (marking functions) for the stable end-to-end operation over the Internet. However, there is a significant fraction of uncontrolled flows such as real-time video and audio flows in the Internet, and the effect of those uncontrolled flows on the design and simulation of the Internet cannot be ignored in the use of fluid models.

In this thesis, we first explore the use of time-scale decomposition of the end-systems and queueing dynamics at the intermediate routers. Based on this time-scale decomposition study, for a queue-based router model we develop an equivalent fluid model that depends only on the instantaneous traffic rate. The main intuition for such a rate based model is that there exists a sufficient randomness in the Internet due to uncontrolled flows. We next study how the rate based model can be practically used for a Internet simulation/emulation with a mixture of controlled and uncontrolled flows. We address this by

developing a hybrid network simulator – FluNet – which combines actual network hardware (routers and switches) at the network edge, and rate based fluid models within the network core.

Second, we study the effects of these uncontrolled flows on the design choices for the end controllers and marking function. Current research has focused on the design of controllers and network algorithms with the objective of stability and convergence of the transmission rates. However, an important criterion that has not received much attention is the design of the controllers with the objective of providing QoS guarantees to real-time flows that share the links with the controlled flows. In this thesis, we study the design rules for network congestion controllers with the objective of providing QoS support for such real-time flows.

Third, while much of the research based on fluid models has focused on wireline networks, a growing area of research is that of wireless multi-hop networks. The main issue in using fluid models in this context is that the MAC (media access control) is a "discrete" entity, as a result of which fluid models have not been commonly used. In this work, we first investigate fluid models for MAC and appropriate models for congestion control over multi-hop wireless networks. Based on an optimization framework with constraints that arise from the multi-hop wireless network, we propose hop-by-hop congestion control algorithms, and study their properties on the stability and peak buffer requirement.

Fourth, we study a realistic MAC protocol, which leads to the appropriate fluid models used for analysis and design of networking algorithms (i.e., congestion control) over wireless multi-hop networks. In this work, we study a distributed randomized MAC protocol that converges an optimal schedule with a simple one-hop synchronized contention signaling mechanism. Furthermore, by simulation and analysis, we show that the proposed protocol adapts well to slowly-varying load/topology changes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the past decade, the Internet has experienced tremendous growth in scale, speed, and heterogeneity, and the control and management of the Internet is becoming an ever more important issue. With this trend, understanding the behavior of large-scale networks is crucial for the future success of the Internet. Analyzing and predicting the performance of large-scale network systems requires: (i) good analytical models to capture the dynamics of the systems correctly, and/or (ii) large-scale simulation setup or a network test-bed where the systems can be studied in a controlled environment.

However, simulation of large-scale network systems with thousands of users and flows passing over large numbers of routers with complex routing patterns is very difficult due to computational complexity. Further, having a large hardware test-bed over which we can actually deploy algorithms can be extremely expensive to implement. On the other hand, exact mathematical modeling of complex and large-scale networks seems analytically intractable.

To address these issues, fluid model based approximations, which consider traffic behavior in terms of packet rates rather than packet instances, have been widely used in literature and have been shown to be efficient and accurate in the modeling, analysis [1–11], and simulation [12–20] of the Internet. From a modeling and analysis perspective, much of the research has focused on the design of end host controllers and control algorithms at routers (marking functions) for the stable end-to-end operation of controlled flows over the

Internet. These studies mostly ignore the effect of and the effect on the uncontrolled flows[1], which constitute a significant fraction of the Internet traffic. Similarly, research on fluid models based simulation also ignores the effect of uncontrolled flows. Thus, in the first part of this thesis, we study new fluid models which explicitly consider uncontrolled flows, and their relevance in modeling, analysis, and simulation/emulation of the Internet. Further, we also study the effects of uncontrolled flows on the design choices for the end-host controllers and the queue management algorithm at the intermediate routers.

While much of the research based on fluid models has focused on wire-line end-to-end controllers, a growing area of research is that over wireless networks. The second part of this thesis focuses on wireless multi-hop networks. A wireless multi-hop network is a system of network hosts (nodes) connected by wireless links. An example of a multi-hop wireless network is a community based roof-top network, where a collection of Base Stations (BS) mounted on roof-tops collaborate to relay data packets. In other words, if two hosts are not within radio range, all communication messages between them must pass through one or more intermediate hosts that act as routers.

The communication patterns and resource sharing mechanisms over such a network are significantly different from those of a wired network, mainly due to "coupling" of simultaneous transmissions imposed by the nature of the wireless channel (e.g., interference constraints and packet collisions among the nearby transmissions), thus leading to different approaches to analysis and design of network algorithms. This "transmission coupling" in a wireless network leads to a tightly coupled resource allocation problem, not only across nodes, but also across multiples layers (e.g., transport layer congestion control and the Media Access Control (MAC) layer scheduling). The primary issue in using fluid models is that the MAC is a "discrete" entity, and is not easily amenable to a fluid model. In this thesis, we first study fluid models for MAC and the corresponding models for congestion control over multi-hop wireless networks. Next, we study optimal MAC designs that lead to these fluid models over a longer time-scale.

[1]We use the terminology "controlled flows" to refer to flows of data traffic which react and adapt their transmission rates in real-time to feedback from the network. An example of such a flow is a TCP (Transmission Control Protocol) flow. "Uncontrolled flows" refer to data flows that do not react to network feedback. Examples of such flows include real time video/audio as well as web mice – short bursts of packet generated by web requests. Uncontrolled flows are also called unresponsive flows in the literature.

1.1 Main Contributions and Organization

In Chapter 2, we show that the randomness due to uncontrolled flows in the Internet is sufficient to decouple the dynamics of the router queues from those of the end controllers. This implies that a time-scale decomposition naturally occurs such that the dynamics of the router manifest only through their statistical steady-state behavior. We show that this time-scale decomposition implies that a queue-length based marking function (e.g., RED-like and REM-like algorithms) has an equivalent form which depends only on the data arrival rate from the end-systems and does not depend on the queue dynamics.

In Chapter 3, using the ideas and the equivalent rate-based fluid model in Chapter 2, we propose a hybrid simulator – FluNet – where queueing dynamics are not tracked. The FluNet simulator is predicated on a fast-queueing regime at bottleneck routers, where the queue length fluctuates on a time-scale that is much faster than the time-scale of end systems. FluNet does not track queue lengths at routers, but instead, uses an equivalent rate based model at router queues, and queue-based AQM schemes are replaced by such an equivalent rate-based model. This allows us to simulate large-scale systems, where the simulation "time step-size" is governed only by the time-scale of the end-systems, and not by that of the intermediate routers; whereas a fluid model based simulator that *tracks* queue-length would require decreasingly smaller step-sizes as the system scale size (such as the number of flows and link capacity) increases.

In Chapter 4, we study the effects of marking elasticity (which characterizes how aggressively the marking function responds to congestion) on the QoS for uncontrolled real-time flows, at a router accessed by both uncontrolled and controlled flows. First, we derive lower and upper bounds on the queue overflow probability at a router of a single bottleneck system. Using this, we quantify the trade-off between stability for controlled flows and QoS guarantee for uncontrolled real-time flows as a function of marking elasticity. The results indicate that some marking functions may be "uniformly" better than others. In particular, among the marking functions that we have compared, our bounds indicate that a rate based version of REM seems to provide the largest local-stability region for *any* given QoS requirement. Next, we compare the capacity required at a router with

only FIFO scheduling versus a router with priority scheduling (priority given to the real-time flows) for supporting a given QoS requirement (queue overflow probability). We quantify the "scheduling-gain" (in terms of supporting QoS for the real-time flows) of priority scheduling over FIFO scheduling, as a function of marking elasticity. We show that this scheduling gain decreases with more elastic marking functions. In other words, the difference in the required capacities with FIFO and priority scheduling for a fixed QoS (queue overflow probability) can be significantly reduced by increasing the marking elasticity.

In Chapter 5, we develop a fair hop-by-hop congestion control algorithm with the MAC constraint being imposed in the form of a channel access time constraint, using an optimization based framework. In the absence of delay, we show that this algorithm is globally stable using a Lyapunov function based approach. Next, in the presence of delay, we show that the hop-by-hop control algorithm has the property of spatial spreading. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. We derive bounds on the "peak load" at a node both with hop-by-hop control and with end-to-end control, and show that significant gains are to be had with the hop-by-hop scheme.

In Chapter 6, we note that aggregate traffic loads and topology in multi-hop wireless networks may vary slowly, permitting MAC protocols to 'learn' how to spatially coordinate and adapt contention patterns. Such an approach could reduce contention, leading to better throughput and energy consumption. To that end we propose a new family of distributed TDMA MAC scheduling algorithms combining synchronous two-level priority RTS/CTS handshaking with randomized time slot selection. We prove that for any fixed admissible load such algorithms converge to a feasible schedule exponentially fast, and so are *throughput-optimal*. Furthermore, by adaptively biasing time-slot selection probabilities based on past history, one can develop variations that are also provably throughput-optimal and exhibit better convergence rates. Additionally under moderate loads local changes in load would lead to only local changes in contention patterns leading once again to fast convergence. This makes the case for adopting such protocols in wireless multi-hop networks, where aggregate loads and network topology are slowly varying.

Finally, we conclude the thesis in Chapter 7.

Chapter 2

Time-scale Decomposition and Equivalent Rate Based Marking

2.1 Overview

Differential equation models for Internet congestion control algorithms have been widely used to understand network dynamics and the design of router algorithms. These models use a fluid approximation for user data traffic, and describe the dynamics of the router queue and user adaptation through coupled differential equations. The interaction between the routers and flows occurs through marking, where routers indicate congestion by appropriately marking packets during congestion.

In this chapter, we show that the randomness due to short and unresponsive flows in the Internet is sufficient to decouple the dynamics of the router queues from those of the end controllers. This implies that a time-scale decomposition naturally occurs such that the dynamics of the router manifest only through their *statistical steady-state* behavior. We show that this time-scale decomposition implies that a queue-length based marking function (e.g., RED-like and REM-like algorithms, which have no queue averaging, but depend only on the instantaneous queue length) has an *equivalent* form which depends *only on the data arrival rate from the end-systems and does not depend on the queue dynamics*. This leads to much simpler dynamics of the differential equation models (there is no queueing dynamics to consider), which enables easier analysis and could be potentially used for low

complexity fast simulation.

Using packet based simulations, we study queue based marking schemes and their equivalent rate based marking schemes for different types of controlled sources (proportional fair and TCP) and queue based marking schemes. Our results indicate a good match in the rates observed at the intermediate router with the queue based marking function and the corresponding rate based approximation. Further, the window size distributions of a typical TCP flow with a queue based marking function as well as the equivalent rate based marking function match closely, indicating that replacing a queue based marking function by its equivalent rate based function does not statistically affect the end host's behavior.

2.2 Introduction

We consider the problem of Internet congestion control when the network is accessed by a mixture of long-lived controlled flows, as well as short-flows which do not react to congestion. The short flows model a mixture of real-time based traffic (such as real-time multimedia) as well as web traffic (so called web-mice), where the sessions are too short for the end systems to react to network congestion.

The transmission rate of the long-lived flows are controlled by the intermediate routers in the network. The task of these routers is to simply notify the end systems whenever they detect congestion in the network. Associated with each router is a marking function, which *marks* a fraction of the flow, and the fraction that is marked is a function of the arrival rate (rate based marking) or the queue length (queue based marking). In the Internet, marking is implemented via the Explicit Congestion Notification (ECN) mechanism [21], where packets have a bit in the header that can be set to '1' to indicate congestion. The end-host reacts to this information by suitably adapting its transmission rate, thus adapting to network congestion.

There has been extensive research on differential equation based congestion control [12,22–26], where fluid models of a large number of flows were used to model the dynamics of the system based on a rate based marking scheme. The source controllers are modeled by differential equations (i.e., a fluid model model for data flow). These controllers adapt

their transmission rate based on network feedback in the form of a fraction of fluid that is marked by the routers. In other words, with n flows in the network, the dynamics of the controller are described by

$$\dot{x}_n^i(t) = \kappa\Big(w - U_i'(x_n^i(t))^{-1} p_r\Big(\frac{1}{n}\sum_{j=1}^{n}(a_n^j(t) + x_n^j(t))\Big)\Big),\ i = 1, \cdots, n, \tag{2.1}$$

where w, κ are parameters of the controller that determine the equilibrium rate as well as the transient dynamics. $x_n^i(t)$ is the transmission rate of the controlled flow i at time t, and $\sum_i a_n^i(t)$ represents the short-lived uncontrolled flows. $U_i(x)$ is an concave utility function of the user i, when the transmission rate of the user i is x. Examples of such utility functions include $\log x$ (proportional fair controller) and $-1/x$ (TCP controller) [2]. The function $p_r(\cdot)$ is a *rate based marking* function whose argument is *the average arrival rate to the router* (additional discussion is available later in this section). The marking function indicates the level of congestion at the router. Thus, $p_r(\cdot)$ is a monotone, increasing function with range $[0, 1]$. The larger the marking level is, the higher is the perceived congestion at the router. As seen in (2.1), the controller reacts to a congestion level by decreasing the transmission rate.

Alternately, instead of adapting based on the average arrival rate, the marking function at the router can adapt based on the *queue length at the router.* In other words, the router is associated with a *queue based marking* function $p_q(\cdot)$. This is assumed to be a monotone increasing function over $[0, 1]$, and Lipschitz continuous with parameter L_q. The associated differential equation model for the end-system controller is given by

$$\dot{x}_n^i(t) = \kappa\left(w - U_i'(x_n^i(t))^{-1} p_q\Big(\frac{1}{n}Q_n(t)\Big)\right), \tag{2.2}$$

where

$$\dot{Q}_n(t) = \begin{cases} \displaystyle\sum_{j=1}^{n}\Big(a_n^j(t) + x_n^j(t)\Big) - nc & \text{if } Q_n(t) > 0, \\ \left[\displaystyle\sum_{j=1}^{n}\Big(a_n^j(t) + x_n^j(t)\Big) - nc\right]^+ & \text{if } Q_n(t) = 0. \end{cases}$$

$Q_n(t)$ is the queue length at the router, and nc is the capacity of the link. Examples of queue based marking include Random Early Discard (RED) [21], Adaptive Virtual Queue (AVQ) [27], and Random Exponential Marking (REM) [28].

It has been shown in [26] that the differential equation based models described in (2.1) and (2.2) are valid models of in the Internet when there are large enough number of flows and the network capacity is large (scaled with the number of flows). In such a regime, the arguments of the marking functions are interpreted as the *average arrival rate* (averaging by the number of flows) or the *scaled queue length* (scaled by the number of flows) respectively. Physically, this scaling of the arguments correspond to the fact that the arrival rates and capacity are large, see [26] for details.

In other words, for a network model with n flows and the capacity at the router being nc, the marking function at the router adapts either based on the average arrival rate $x(t) = \frac{1}{n}X^n(t)$, where $X^n(t)$ is the total arrival rate to the router, or based on the average queue length $q(t) = \frac{1}{n}Q^n(t)$, where $Q^n(t)$ is the queue length at the router. In particular, this implies that *as the system size becomes larger, so does the associated queue length at the router*. In other words, a finite non-zero queue length in the fluid differential equation model (2.2) indicates that the *actual* queue length in the router is large (of order n). Related work with a similar scaling (large buffer and capacity) for window based control is available in [29].

However, as link speeds in modern and future communication networks is becoming higher, high-speed memory buffer with high cost is required in the design of such networks. Therefore, it is questionable if the queue buffers at intermediate routers need to *scale linearly* with the number of flows [30]. In [30, 31], the authors have in fact shown that buffers need not scale with the link speed in order to achieve significant multiplexing gains.

In this chapter, we focus on this regime where the queue length *does not scale with the number of flows*. Such a behavior occurs, for instance, if the queue based marking function $p_q(\cdot)$ is invariant with the number of flows and *is a function of the actual queue length, not the average queue length.* Under such a regime, the queue dynamics occur on a much *faster* time-scale than that of the end system controller [4]. In this context, it is reasonable to expect that queueing dynamics are not visible to the end system controller. Instead, the queueing behavior at the router affects the end system controller only through the statistical

behavior of the queue.

Recent related work includes [32], where the authors consider a discrete time framework for congestion control. They have shown that depending on the scaling, the limiting system could be a combination of queue and rate based marking, even if the unscaled system consists of only queue based marking. In our paper, we consider a continuous time framework, where we are primarily interested in a pure rate based approximation to a queue based marking function. Our focus is on deriving the equivalent rate based marking function over a continuous-time framework, and studying network dynamics by replacing a queue based marking function (such as RED or REM) with an equivalent rate based one. Further, the proof techniques employed are very different in the two approaches.

2.2.1 Main Contributions and Organization

The main contributions of this chapter are the following:

(i) This chapter quantifies the heuristics based on time-scale separation by showing that under suitable assumptions, *queue based marking (based on instantaneous queue length) and the associated queueing dynamics can be approximated by a rate based marking function given by*

$$p(x) \quad \triangleq \quad E_{\pi_{\lambda}^{c-x}}[p_q(Q)],$$

where π_{λ}^{μ} is the stationary queue-length distribution of an M/D/1 queue with Poisson arrival rate λ and capacity μ. The parameter x and λ is simply the average arrival rate from the controlled and the uncontrolled flows (averaging over flows, not time) to the router queue, respectively.

(ii) Using packet simulations by suitably modifying the *ns-2* [33] simulator, we compare queue based marking schemes and their equivalent rate based marking schemes for different types of controlled sources (proportional fair and TCP) and queue based marking schemes (RED-like and REM-like algorithms without queue averaging). In addition, we show that the equivalent rate based marking scheme behaves well even when drastic changes occur in the number of network connections. The simulation

results indicate a good match between the queue based marking and the equivalent rate based marking in the steady-states as well as the transient behaviors of end-sources.

The results of this chapter potentially could enable low complexity simulation and easier analysis for the following reasons. In the context of network simulation, several widely-used discrete event-driven simulators are available [33–36]. However, with a large number of flows the number of events at an intermediate router scales with the number of flows due to packet arrivals/departures and marking computation events. The equivalent rate based model proposed in this chapter uses a marking averaged by rates, along with the absence of queuing dynamics to enable a simulation complexity at intermediate routers that does not scale with the number of flows, leading to significant reduction of simulation complexity (see Section 2.4.3 for additional discussion).

Also, much of the work on stability analysis of congestion control algorithm has been done based on the rate based marking model at the intermediate routers. Thus, with our approximate rate based model, it seems easier to study queue based marking systems using analytical tools developed for rate based marking in literature [7, 24, 26].

In the rest of this chapter, we begin with a description of the system model in Section 2.3. Next, in Section 2.4, we show that there exists an equivalent rate based marking function for a given queue based marking scheme (under suitable conditions). Using these results, we derive expressions for the equivalent marking function with the RED-like and the REM-like controllers[1] in section 2.4.4. We finally present simulation results for RED and REM with proportional fair and TCP sources.

2.3 System Model

Consider the system shown in Figure 2.1. We consider a single queue with the FIFO (First In First Out) scheduling discipline accessed by two types of flows: *(i)* controlled flows and *(ii)* uncontrolled flows. We consider a sequence of systems indexed by n, the scaling parameter. In the n-th system, the queue is fed by n independent, identically distributed uncontrolled

[1] Henceforth, for notional convenience we use the terms RED and REM to refer to queue based RED and REM without queue averaging.

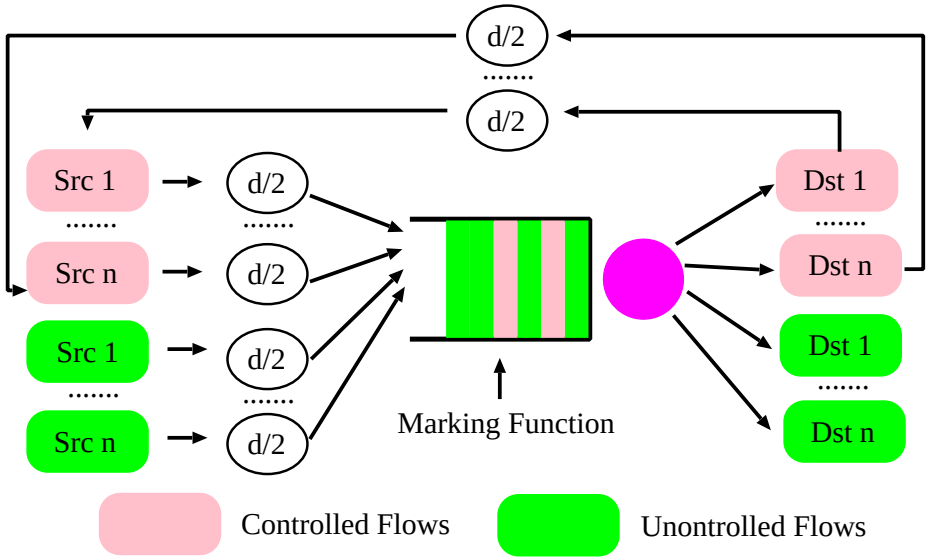


Figure 2.1: System model

flows and by n controlled flows determined by a congestion control algorithm[2]. The output capacity of the router queue scaled with n as nc pkts/sec.

For the n-th system, we model each uncontrolled flow by means of a point process $A_n^i(t)$, that represents the cumulative number of packets from flow i that arrive until time t. We assume that each $A_n^i(t)$ has the same distribution as a simple stationary point process A that satisfies the following assumptions [30, 37].

Assumption 2.3.1. *A is a simple stationary point process satisfying the following three properties.*

(i) There exists $\lambda > 0$ such that $E[A(t)] = \lambda t$ for $t \in [0, \infty)$.

(ii) There exists $\theta_0 > 0$ and $K < \infty$ such that

$$\lim_{t \to 0^+} E[e^{\theta_0 A(t)} \mathbf{1}_{A(t)>K}] \quad = \quad 0,$$

where $\mathbf{1}_E = 1$ if the predicate E is true, and 0 otherwise.

(iii) $\liminf_{t\to\infty} \frac{t\Lambda(x,t)}{\log t} > 0$, *where* $\Lambda(x,t) = \sup_{\theta \in \mathcal{R}}[\theta x - \frac{1}{t} \log E[e^{\theta A(t)}]]$.

[2]For notational simplicity, we have assumed an equal number of controlled and uncontrolled sources. The results in this paper hold even if they are not the same, as long as the ratio of the number of controlled flows and the number of uncontrolled flows is finite.

Assumption 2.3.1 states that each uncontrolled arrival process satisfies the properties that *(i)* multiple packets from a single uncontrolled source do not arrive at the same time, *(ii)* all arriving packets are of the same size and *(iii)* the uncontrolled arrival process has a finite intensity (see [30] for further details).

From the controlled flows point of view, the system we have described above can be thought of as a closed loop system with delay, and feedback control applied at the routers based on *queue based marking function* denoted by $p_q(\cdot)$. A popular modeling and analysis methodology for such closed-loop systems in the Internet context has been through functional differential equations based *fluid models*.

The generic model of such a system consists of a collection of user flows, a router modeled by marking functions which signal congestion by marking flows, and receivers which detect the marks and informs the respective flows to increase or decrease their transmission rate. We model flows by fluid processes. We denote the fluid rates of individual flows in the n-th system by $\{x_n^i(t), i = 1, \ldots, n\}$, where $x_n^i(t)$ denotes the transmission rate of a controlled flow i at time t. The dynamics of the transmission rate for each user are governed by a differential equation based controller as discussed in Section 2.2. We comment that the controller in (2.2) is called a proportional-fair controller if $U(x) = \log(x)$ [4], as controllers of this form lead to a proportionally-fair allocation of bandwidth across users. The results in this chapter, however, apply to any differential equation based congestion controller as long as $\dot{x}_n^i(\cdot)$ is bounded (i.e., the transmission rate is Lipschitz). In particular, suppose that the transmission rate $x_n^i(\cdot)$ is bounded by some constant L. This in-turn implies that $x_n^i(\cdot)$ is Lipschitz continuous with some parameter $M < \infty$ [9]. In the rest of this chapter, we assume that the transmission rate is Lipschitz continuous with parameter M.

Let $A_n(t) = \sum_i A_n^i(t)$ be the *cumulative number of arrivals* until time t due to *uncontrolled flows,* and $X_n(t) = \sum_i x_n^i(t)$ be the total *arrival rate* at time t due to controlled flows. From Assumption 2.3.1, $E(A_n(t)) = n\lambda t$.

For the *controlled flows,* let us denote the average arrival rate by

$$x_n(t) \quad = \quad \frac{1}{n} X_n(t).$$

Further, we define the total volume of arrivals (due to the controlled flows) until time t by $Y_n(t)$, where

$$Y_n(t) \;=\; \int_0^t X_n(z)dz \;=\; n\int_0^t x_n(z)dz$$

Finally, we assume that the initial conditions satisfy

$$\begin{aligned} x_n^i(0) &\xrightarrow{n\to\infty} x^i(0) \\ x_n(0) &\xrightarrow{n\to\infty} x(0) \\ Q_n(0) &\xrightarrow{n\to\infty} Q(0) < \infty \\ x(0)+\lambda &< c \end{aligned} \tag{2.3}$$

Heuristically, these conditions correspond to the assumption that the initial condition is well defined, and is a stable system.

2.4 Limiting Rate Based Marking Function

In this section, we will derive the equivalent rate based marking function for a given queue based marking function. In this chapter, we focus on the instantaneous queue length process. Note that popular AQM algorithms such as RED and REM use (exponentially moving) average queue length to mark the incoming packets. We left the study on AQM algorithms with queue averaging as future work.

For a fixed $T > 0$, we are interested in studying the queue length process (which measures the volume of data at the router), denoted by $Q_n(t)$, over the time-interval $[0, \frac{T}{n}]$. Thus, we are interested in the queueing behavior at the router over a short interval of time. Even over this small time interval, we will show that the queue reaches "steady-state" behavior. This occurs due to the fact that the capacity is very large (nc), and causes the queue to "regenerate" an arbitrarily large number of times over the interval $[0, \frac{T}{n}]$.

However, from a single *end-system* (the user) point of view, this corresponds to a very short interval of time. Thus, one can expect that the end-user will only perceive the statistical "steady-state" queueing behavior. The results in this section quantify the above

heuristic.

For any $s \in [0, \frac{T}{n}]$, the queue length process is given by

$$\begin{aligned} Q_n(s) &= \sup_{r\in[0,s]} \Big[A_n(s) - A_n(r) + Y_n(s) - Y_n(r) - nc(s-r) + Q_n(r)\Big] \\ &= \sup_{r\in[0,s]} \Big[A_n(s) - A_n(r) + n\int_r^s x_n(z)\,dz - nc(s-r) + Q_n(r)\Big] \end{aligned}$$

Now, let us study the processes (X_n, Y_n, A_n, Q_n) over a *slowed-down* time-scale. In other words, for $t \in [0, T]$, we define the processes

$$q_n(t) \triangleq Q_n\left(\frac{t}{n}\right), \quad a_n(t) \triangleq A_n\left(\frac{t}{n}\right), \quad y_n(t) \triangleq Y_n\left(\frac{t}{n}\right)$$

Thus, we have for any $t \in [0, T]$,

$$\begin{aligned} q_n(t) &= Q_n\Big(\frac{t}{n}\Big) \\ &= \sup_{\frac{r}{n}\in[0,\frac{t}{n}]} \Big[A_n\left(\frac{t}{n}\right) - A_n\left(\frac{r}{n}\right) + Y_n\left(\frac{t}{n}\right) - Y_n\left(\frac{r}{n}\right) - \frac{nc(t-r)}{n} + Q_n\left(\frac{r}{n}\right)\Big] \\ &= \sup_{r\in[0,t]} [a_n(t) - a_n(r) + y_n(t) - y_n(r) - c(t-r) + q_n(r)] \end{aligned} \tag{2.4}$$

By assumption, each individual data rate $(x_n^i(t))$ is Lipschitz continuous with some parameter $M < \infty$. This also implies that the average data rate $(x_n(r))$ is Lipschitz continuous with parameter M. Let us now define

$$\tilde{q}_n(t) \triangleq \sup_{r\in[0,t]} [a_n(t) - a_n(r) + (t-r)x(0) - c(t-r) + \tilde{q}_n(r)] \tag{2.5}$$

2.4.1 Convergence of the Queue length Trajectory

We now show that the queue length process over the slowed-down time-scale converges weakly to the queue length process of a M/D/1 queue with service rate $c - x(0)$. In [30], the authors showed a similar result for the stationary distribution of the queue. In this paper, we are interested in the *path properties of the queue* because the marks received by the end-user depends on the integral of the marking function over the (unscaled) time interval $[0, T/n]$. Thus, it is not sufficient for us to consider only the stationary distribution. We

show that the slowed-down queue length process converges to the corresponding M/D/1 queueing process "uniformly" (to be precise, with respect to the Skorohod metric) over the time interval $[0, T]$.

Prior to presenting the main theorems, we first provide the following two lemmas.

Lemma 2.4.1. *Given $\epsilon > 0$, we can find N such that $\forall n > N$,*

$$\|q_n(t) - \tilde{q}_n(t)\| \quad < \quad \epsilon, \tag{2.6}$$

where $\|\cdot\|$ is the Skorohod metric [38] in the space $\mathcal{D}([0, T] : \mathcal{R}^+)$.

Proof. The proof is presented in the Appendix. □

Lemma 2.4.2. *Suppose that $a_n(t) \to a(t)$ in the space $\mathcal{D}([0, T] : \mathcal{R}^+)$. Then, given any $\epsilon > 0$, there exists N such that $\forall n > N$ we have*

$$\|q(t) - \tilde{q}_n(t)\| \quad < \quad \epsilon, \tag{2.7}$$

where $q(t)$ is defined by

$$q(t) \quad \triangleq \quad \sup_{r \in [0,t]} [a(t) - a(r) + (t - r)x(0) - c(t - r) + q(r)], \tag{2.8}$$

and $a(t)$ is a Poisson process with arrival rate λ.

Proof. The proof is presented in the Appendix. □

Theorem 2.4.1. *As $n \to \infty$, we have*

$$q_n(t) \quad \xrightarrow{w} \quad q(t), \quad t \in [0, T] \quad over \quad \mathcal{D}([0, T] : \mathcal{R}^+)$$

where $\xrightarrow{w}$ represents weak convergence, and $q(t)$ is the queue-length process of a single server M/D/1 queue, with deterministic service rate $c - x(0)$, and arrival process $a(t)$, which is a Poisson process of rate λ.

Proof. From the superposition theorem for point processes [37], we know that a_n converges weakly to a Poisson process with rate λ denoted by $a(t)$ in $\mathcal{D}([0, T] : \mathcal{R}^+)$. From the Skorohod

representation theorem [38], we can find processes $a'_n(t)$ and $a'(t)$ in $\mathcal{D}([0,T]:\mathcal{R}^+)$ such that

$$\begin{aligned} a_n(t) &\overset{\text{dist}}{=} a'_n(t) \\ a(t) &\overset{\text{dist}}{=} a'(t), \end{aligned}$$

where $\overset{\text{dist}}{=}$ means "equivalence in distribution" and

$$\|a'_n(t) - a'(t)\| \xrightarrow{n\to\infty} 0 \quad \text{in} \quad \mathcal{D}([0,T]:\mathcal{R}^+) \tag{2.9}$$

Corresponding to the arrival processes $a'(t)$ and $a'_n(t)$, let us define $q(t)$, $q'_n(t)$, and $\tilde{q}'_n(t)$ by Equations (2.4), (2.5), and (2.8) respectively.

Then, it suffices to prove that $\forall \epsilon > 0$, we can find N such that $\forall n > N, \|q'_n(t) - q'(t)\| < \epsilon$ in the space $\mathcal{D}([0,T]:\mathcal{R}^+)$. By the triangle inequality of Skorohod norm, we have

$$\|q'_n(t) - q'(t)\| \leq \|q'_n(t) - \tilde{q}'_n(t)\| + \|\tilde{q}'_n(t) - q'(t)\|$$

By applying Lemma 2.4.1 to the first term of RHS and Lemma 2.4.2 to the second term of RHS, the result follows. □

Using this result, we now show that the total volume of marks received over the (slowed-down) time-interval $[0,T]$ converges that given by an M/D/1 queue.

Theorem 2.4.2. *Suppose that*

$$q_n(t) \xrightarrow{w} q(t), \quad t \in [0,T] \quad \textit{over} \quad \mathcal{D}([0,T]:\mathcal{R}^+),$$

where $q_n(t)$ and $q(t)$ is defined as (2.4) and (2.8). Then, we have

$$\int_0^T p_q(q_n(y))\,dy \xrightarrow{w} \int_0^T p_q(q(y))\,dy \tag{2.10}$$

$$\int_0^T x_n^i\left(\frac{y}{n}\right) p_q(q_n(y))\,dy \xrightarrow{w} \int_0^T x^i(0) p_q(q(y))\,dy \tag{2.11}$$

Proof. From Theorem 2.4.1 and Skorohod representation theorem, we can find q'_n and q' in $\mathcal{D}([0,T]:\mathcal{R}^+)$ such that q'_n converges to q' in the Skorohod topology. By the definition

of convergence in the Skorohod topology, we can find a strictly increasing, continuous function λ_n of $[0,T]$ onto itself and $N_1 > 0$ such that for a given $\epsilon > 0$, and $\forall n > N_1$,

$$\begin{aligned} \sup_{t\in[0,T]} |q_n'(\lambda_n(t)) - q'(t)| &< \epsilon \\ \sup_{t\in[0,T]} |\lambda_n(t) - t| &< \epsilon \end{aligned} \tag{2.12}$$

By adding and subtracting a common term, we have

$$\begin{aligned} \left|\int_0^T p_q(q_n'(y))\,dy - \int_0^T p_q(q'(y))\,dy\right| &\leq \left|\int_0^T p_q(q_n'(y))\,dy - \int_0^T p_q(q_n'(\lambda_n(y)))\,dy\right| \\ &+ \left|\int_0^T p_q(q_n'(\lambda_n(y)))\,dy - \int_0^T p_q(q'(y))\,dy\right| \end{aligned}$$

For the second term of RHS, using Lipschitz continuity assumption of p_q and the condition (2.12),

$$\left|\int_0^T p_q(q_n'(\lambda_n(y)))\,dy - \int_0^T p_q(q'(y))\,dy\right| \leq \int_0^T L_q|q_n'(\lambda_n(y)) - q'(y)| < L_q T\epsilon, \tag{2.13}$$

where $0 < L_q < \infty$ is the Lipschitz constant of $p_q(\cdot)$.

Next, for the first term of RHS, We know that $q'(s) \in \mathcal{D}([0,T] : \mathcal{R}^+)$ has a finite number of jumps denoted by $\mathcal{J}(q') < \infty$, since the arrival process is a Poisson process with a finite rate over the finite interval of time $[0,T]$. From the condition (2.12), we can find $N_2 > 0$ such that $\forall n > N_2$, $J \triangleq \mathcal{J}(q_n') = \mathcal{J}(q')$. Let us denote the jump times of $q_n'(s)$ by $\{t_n^j, j = 1,\ldots,J\}$.

Now, we divide the entire interval $[0,T]$ into two sets of intervals A_1 and A_2, where $A_1 = \{I_j \triangleq [t_n^j - \epsilon, t_n^j + \epsilon], j = 1,\ldots,J\}$ and $A_2 = [0,T]\backslash A_1$. By taking $\epsilon < 0.5\min\{t_n^1, t_n^2 - t_n^1, \ldots, T - t_n^J\}$, this ensures that there is only one jump of the processes, $q_n'(\lambda_n(s))$ and $q_n'(s)$ in the interval I_j. From Lipschitz continuity of p_q, $\forall s \in [0,T]$,

$$\left|p_q(q_n'(\lambda_n(s))) - p_q(q_n'(s))\right| \leq L_q|q_n'(\lambda_n(s)) - q_n'(s)|$$

Let $N_{max} = \max(N_1, N_2)$. Then, $\forall n > N_{max}$,

$$|q_n'(\lambda_n(s)) - q_n'(s)| \quad \leq \quad \begin{cases} 1 & \text{if} \quad s \in A_1, \\ \epsilon(c - x(0)) & \text{if} \quad s \in A_2. \end{cases}$$

Thus,

$$\begin{aligned} & \left| \int_0^T p_q\left(q_n'(y)\right)\, dy - \int_0^T p_q\left(q_n'(\lambda_n(y))\right)\, dy \right| \\ \leq \; & \left| \int_{A_1} p_q\left(q_n'(y)\right)\, dy - \int_{A_1} p_q\left(q_n'(\lambda_n(y))\right)\, dy \right| + \left| \int_{A_2} p_q\left(q_n'(y)\right)\, dy - \int_{A_2} p_q\left(q_n'(\lambda_n(y))\right)\, dy \right| \\ \leq \; & \int_{A_1} \left| p_q\left(q_n'(y)\right) - p_q(q_n'(\lambda_n(y))) \right| dy + \int_{A_2} \left| p_q\left(q_n'(y)\right) - p_q(q_n'(\lambda_n(y))) \right| dy \\ \leq \; & 2L_q J\epsilon + L_q(c - x(0))\epsilon(T - 2J\epsilon) \\ = \; & \epsilon\left(2L_q J + L_q(c - x(0))(T - 2J\epsilon)\right) \end{aligned} \tag{2.14}$$

Since ϵ is arbitrary in (2.13) and (2.14), this completes the proof. The proof of (2.11) is analogous. □

2.4.2 An Equivalent Rate Based Marking Function

This section defines an equivalent rate based marking function based on Theorem 2.4.2 and Theorem 2.4.1. Let us consider the marks received over the time-interval $[0, \frac{T}{n}]$ by some user i. By definition, *the marked volume of data* over this time-interval is given by

$$\int_0^{\frac{T}{n}} x_n^i(y) p_q(Q_n(y))\, dy \quad = \quad \frac{1}{n} \int_0^T x_n^i(y/n) p_q(q_n(y))\, dy$$

Thus, *the time-average volume of marks* received by user i over the time-interval $[0, \frac{T}{n}]$ is

$$\frac{1}{T/n} \int_0^{\frac{T}{n}} x_n^i(y) p_q(Q_n(y))\, dy \quad = \quad \frac{1}{T} \int_0^T x_n^i(\frac{y}{n}) p_q(q_n(y))\, dy \tag{2.15}$$

Thus, from Theorem 2.4.2, we have

$$\frac{1}{T/n} \int_0^{\frac{T}{n}} x_n^i(y) p_q(Q_n(y))\, dy \quad \stackrel{n \to \infty}{\longrightarrow} \quad x^i(0) \frac{1}{T} \int_0^T p_q(q(y))\, dy, \tag{2.16}$$

where $q(y)$ is the queue-length process of an M/D/1 queue with Poisson arrival rate λ and capacity $c - x(0)$. Let us define

$$p_T(x(0)) \triangleq \frac{1}{T}\int_0^T p_q(q(y))\,dy. \tag{2.17}$$

For n large enough, we see from (2.16) that the interaction between the router queuing process and the congestion controller at a fixed user occurs *only through* this function $p_T(\cdot)$.

Further, we observe that $q(y)$ is a regenerative process when $\frac{\lambda}{c-x} < 1$ and $x < c$. Thus, from the ergodic theorem for a regenerative process [39] and Smith's theorem [40], for a given $\epsilon > 0$, $\exists T_0$ such that $\forall T > T_0$,

$$\left|\frac{1}{T}\int_0^T p_q(q(y))\,dy - E_{\pi_\lambda^{c-x}}[p_q(Q)]\right| < \epsilon \tag{2.18}$$

where π_λ^μ is the stationary distribution of an M/D/1 queue with arrival rate λ and capacity μ.

For T large enough, and by defining

$$p(x) = \begin{cases} E_{\pi_\lambda^{c-x}}[p_q(Q)] & \text{if } \frac{\lambda}{c-x} < 1 \text{ and } \quad x < c, \\ 1 & \text{if } \quad x \geq c \text{ or } \frac{\lambda}{c-x} \geq 1. \end{cases} \tag{2.19}$$

we see from (2.17) and (2.18) that the congestion controller dynamics with a queue based marking function $p_q(\cdot)$ can be well approximated by an *equivalent* system with only a rate based controller $p(x)$ at the router, where x is simply the average arrival rate from the controlled flows (averaging over flows, not time) to the router queue.

Remark 2.4.1. *We comment that for each* fixed *T, as $n \to \infty$, the limiting approximate model (2.19) holds. Thus, for T large enough (but finite), we can be within an ϵ bound of the time-averaged limit ((2.18)). Physically, this corresponds to the time-scale separation issue. In reality, we are interested in using this approximation for a finite n (number of flows). Thus, the time-scale T should be chosen such that the following two properties hold:* (i) *The time interval T is large enough such that the randomness due to uncontrolled flows enables the "law of large numbers" to hold (i.e, the M/D/1 queue time-average is close to the stationary distribution), and* (ii) *the time-interval T/n is*

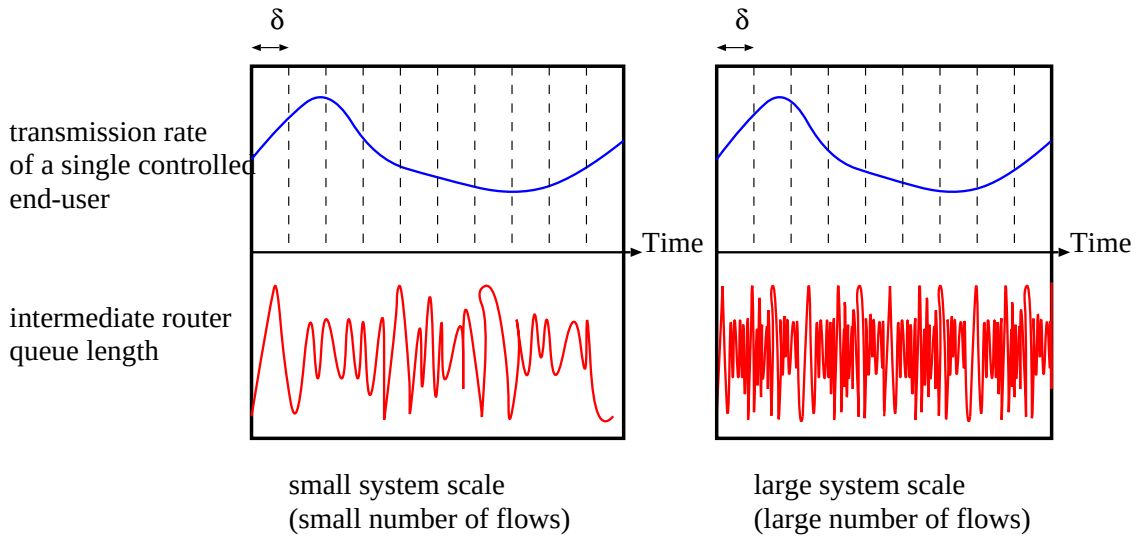


Figure 2.2: System scale size and time step size

small enough such that the arrival rate from a user (controlled flow) does not significantly change. Thus, for any fixed $\epsilon > 0$*, we choose T large enough such that the expected value of the (queue based) marking function with an M/D/1 queue is "close" to the time-average. For this fixed T, we can apply the limit theorem (in n) to justify the rate based approximation.*

If there is insufficient randomness in the network, the value of T could be very large, thus requiring a large value of n (i.e., large number of flows and large capacity) for our analysis to hold. However, our simulations (see Section 2.5) indicate that even with randomness generated due to short ON-OFF flows which occupy about 20%-30% of the link capacity, the value of $n = 100$ *seems to be sufficient, and leads to a match within 5% between a queue based marking function and its equivalent rate based model.*

2.4.3 Application to Simulation Study

Over any fixed interval of time, the simulation complexity at intermediate routers of a queue based simulation depends on the number of events to process. As the number of flows increases, the number of events (packet arrivals/departures, marking probability computation) increases, thus leading to increased simulation complexity. Also, the number of events scales linearly with the number of intermediate routers.

On the other hand, the equivalent rate based model permits the following implementation. Fix a small time-step $\delta > 0$ such that the arrival rate from a controlled flow does not vary significantly over this time-step (i.e., δ is inversely proportional to the end-system congestion controller gain, see Figure 2.2). For this fixed δ, suppose that the number of flows, n, is large enough such that there is a sufficient amount of randomness due to uncontrolled flows over this interval $[0, \delta]$ (i.e., the Poisson approximation holds for the chosen value of δ). Now at each equivalent rate based router, a computation to determine the marking probability is performed only once in each time-step size δ. Such a marking value is computed at each intermediate router, and the packets from end-systems are marked appropriately depending on the marking values. Thus, marking computation required scales as $\frac{1}{\delta} \times n_i$ (n_i is the number of intermediate routers), and is invariant with the number of flows n. Further, such a rate based approximation will become increasingly accurate as the system scale (n) increases.

2.4.4 Examples: REM and RED

In this section, we derive the closed form of equivalent rate based marking functions for the simplified REM and RED controllers, which have no queue averaging, but depend only on the instantaneous queue length.

REM

The simplified version used in this chapter has the following form of the queue based marking function from [28].

$$p_q^{rem}(Q) \quad = \quad 1 - e^{-\alpha Q}, \tag{2.20}$$

where α is a suitable constant pre-defined in the system, and Q is the *queue length* in the system.

First, from the P-K formula for stationary workload V of an M/D/1 queue [41], we have

$$E[e^{-sV}] \quad = \quad \frac{1-\rho}{1-\frac{\lambda}{s}(1-e^{-s/\mu})}, \tag{2.21}$$

where μ is the service rate, λ is the arrival rate, and $\rho = \lambda/\mu$.

Further, for the fluid queueing system we consider, it follows from the definition of workload [41] that $V = Q/c$. Thus, we have

$$\begin{aligned} p(x) &= E_{\pi_\lambda^{c-x}}[p_q^{rem}(Q)] \\ &= E_{\pi_\lambda^{c-x}}[1 - e^{-\alpha c V}] \\ &= 1 - \frac{1-\rho}{1 - \frac{\lambda}{\alpha c}\left(1 - e^{-\alpha c/(c-x)}\right)}, \end{aligned} \tag{2.22}$$

where $\rho = \frac{\lambda}{c-x}$. In the next section, we compare simulation results with queue based marking with REM and compare that to numerical results using its equivalent rate based marking function given by (2.22).

RED

The simplified queue based marking function of RED controller is defined as

$$p_q^{red}(Q) = \max\left(\left(\frac{Q-a}{b}\right)^+, 1\right), \tag{2.23}$$

where a and b are suitable constants pre-defined at the intermediate router [21]. Thus, we have

$$\begin{aligned} p(x) &= E_{\pi_\lambda^{c-x}}[p_q^{red}(Q)] \\ &= E_{\pi_\lambda^{c-x}}\left[\max\left(\left(\frac{Q-a}{b}\right)^+, 1\right)\right] \\ &= \int_a^{a+b}\left(\frac{q-a}{b}\right) f_Q(q)\, dq + \Pr(Q > a+b), \end{aligned} \tag{2.24}$$

In order to evaluate (2.24), it suffices to determine the distribution of the random variable Q. We know that

$$\Pr(Q > y) = \Pr(V > \frac{y}{c})$$

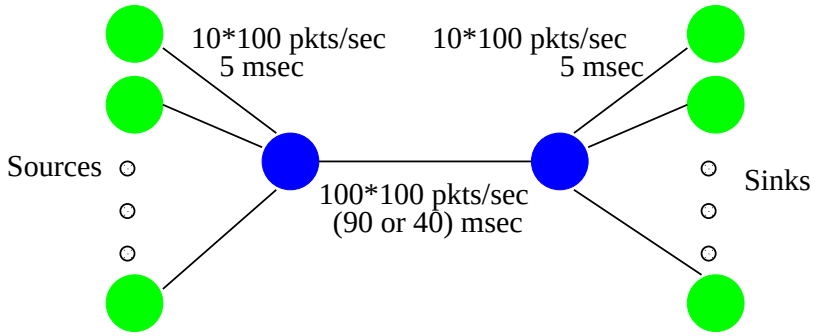


Figure 2.3: Simulation topology

From [42,43], the unfinished work U for a Poisson process with arrival rate λ in the system with service rate μ has a steady state distribution of the form

$$\Pr(U > x) \quad = \quad 1 - (1 - \rho)e^{\rho x} Q_{\lfloor x \rfloor}(x - \lfloor x \rfloor), \tag{2.25}$$

where $\{Q_n(x), n = 0, 1, \cdots\}$ are polynomial functions (which can be calculated recursively as shown in [42,43]), and $\rho = \lambda/\mu$. From the definition of U and V, we have $U = \mu V$. Thus,

$$\Pr(Q > y) \; = \; \Pr(V > \frac{y}{c}) \; = \; \Pr\left(U > \frac{y(c-x)}{c}\right) \tag{2.26}$$

From (2.25) and (2.26), we can evaluate (2.24).

2.5 Simulation

In the previous section, we showed that the queue based marking and the associated queueing dynamics can be approximated by a rate based marking function under the fluid model. In this section, we use the *ns-2* [33] simulator to validate our results. The simulation results in this section show that both the steady-state behavior as well as the transients of the end sources have a good match between the queue based marking and the equivalent rate based marking.

2.5.1 Simulation Environment

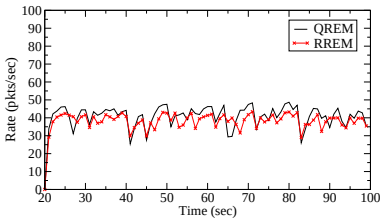
The network topology used in the simulation is shown in Figure 2.3. In Figure 2.3, the bottle-neck link is accessed by 100 controlled and 100 uncontrolled flows and its bandwidth is set to be 100×100 pkts/sec. We set the packet size to be 1000 bytes for controlled and uncontrolled sources in all simulations. The bandwidth and the propagation delay of each access link are set to be 10×100 pkts/sec and 5 ms, respectively. We use 90 msec and 40 msec as the propagation delay of the bottle-neck link (200 msec and 100 msec round-trip delay for the end sources). We assume that the bottle-neck router has only marking functionality, i.e., there is no dropping of the packets due to the buffer overflow (see Section 2.5.2 for additional discussion).

We use the equivalent marking functions for (simplified) RED and REM described in section 2.4.4 as the AQM schemes, and we use two kinds of controlled sources, namely, *(i)* proportional fair controller [4] *(ii)* TCP Sack ([44] suggests the use of TCP Sack or TCP NewReno for network simulation and measurement). Proportional fair controlled source i is described by the following difference equation:

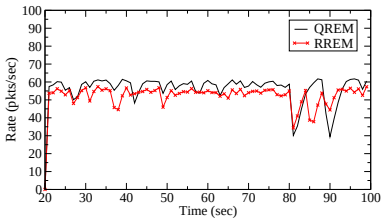
$$x^i[k+1] \quad = \quad x^i_n[k] + u\kappa\left(w - x^i[k-d]p_q\left(Q[k-d]\right)\right),$$

where u is the update interval, and d is the round-tip propagation delay. In our simulation, the update of the source rate is implemented by replacing $x^i_n[k-d]p_q\left(Q_n[k-d]\right)$ with N_k, the actual number of marks received over the update interval u. In our simulations, w and κ are set to be 5.5 and 1, respectively. In addition, we use a value of 200 msec as the update interval. All sources are started with small time differences in order to eliminate synchronization effects between the end-user systems. For notational convenience, we call the queue based RED (REM) as QRED (QREM) and the equivalent rate based marking scheme as RRED (RREM), respectively. In the simulations, we have used the following parameters for RED and REM: $\alpha = 0.02$ and $a = 10, b = 30$.

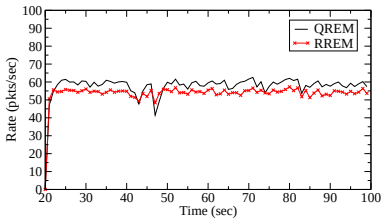
The uncontrolled flows are modeled by ON-OFF processes [45], where the ON and OFF periods are exponentially distributed with parameter 100 msec or 200 msec (denoted by ON-OFF(0.1) and ON-OFF(0.2)) and the packet transmission rate in the ON period is



(a) λ = 50, ON-OFF(0.1), rtt = 200 msec

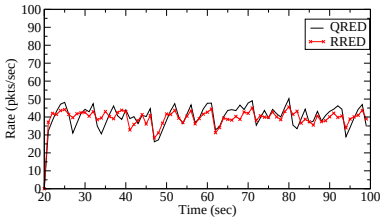


(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

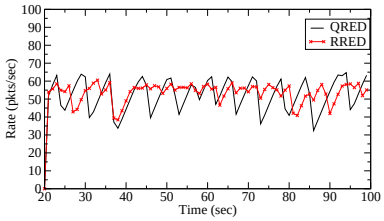


(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

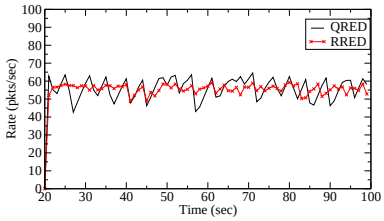
Figure 2.4: Throughput of proportional fair source with REM



(a) λ = 50, ON-OFF(0.1), rtt = 200 msec



(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

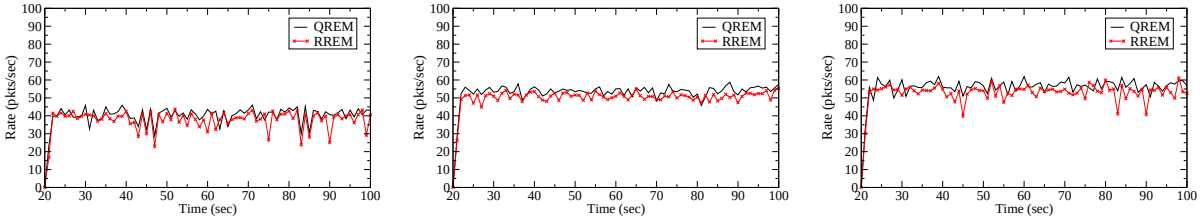


(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

Figure 2.5: Throughput of proportional fair source with RED

suitably set to be constant so that the total load due to the uncontrolled flows is a fixed fraction of the link capacity. We denote the average load due to a uncontrolled flow by λ pkts/sec in the simulation results.

We have two kinds of figures (throughput for both sources and additionally congestion window size for TCP sources) to validate the equivalent rate based marking function proposed in this chapter. In the figures titled "throughput" (see Figures 2.4, 2.6, 2.5, 2.7, and 2.11), we measure and plot the aggregate instantaneous throughput (over all controlled sources) every 0.5 sec, average them over the number of controlled sources, and plot the samples at one second intervals. In addition, we also plot the average rate over (horizontal line in the figures). In the figures for the congestion window size, we trace the instantaneous `cwnd` (congestion window size) value of all TCP sources, average them across the sources, (sampled every 1 second) and plot this as a time series (see Figures 2.8 and 2.9). Finally, we also present the complementary distribution function of the congestion window size (cwnd) for a typical flow under different network conditions (see Figure 2.10).

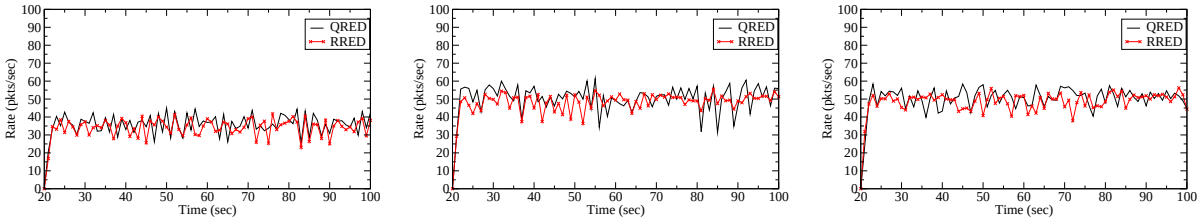


(a) λ = 50, ON-OFF(0.1), rtt = 200 msec

(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

Figure 2.6: Throughput of TCP with REM

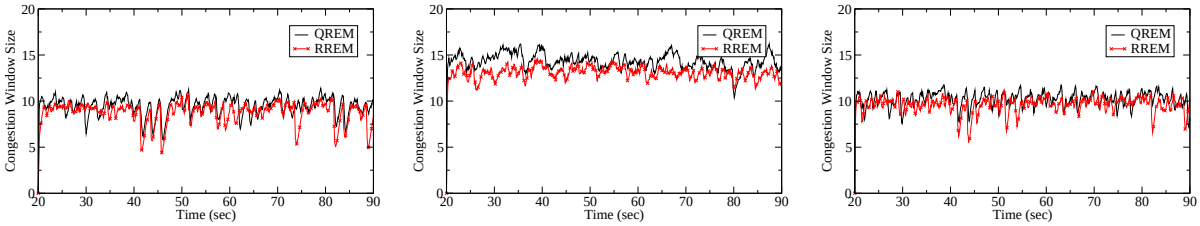


(a) λ = 50, ON-OFF(0.1), rtt = 200 msec

(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

Figure 2.7: Throughput of TCP with RED

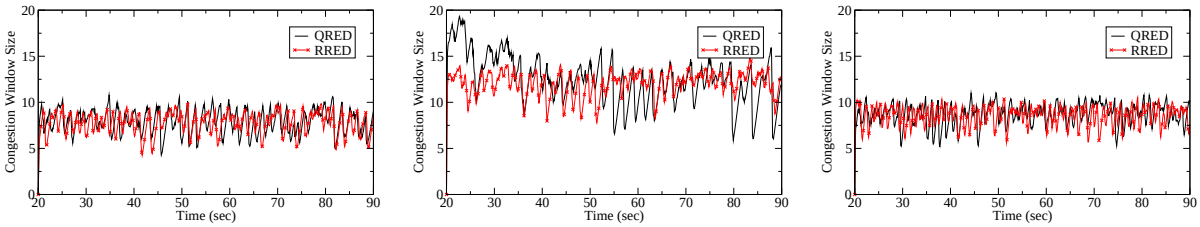


(a) λ = 50, ON-OFF(0.1), rtt = 200 msec

(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

Figure 2.8: Congestion window size of TCP with REM



(a) λ = 50, ON-OFF(0.1), rtt = 200 msec

(b) λ = 35, ON-OFF(0.2), rtt = 200 msec

(c) λ = 35, ON-OFF(0.2), rtt = 100 msec

Figure 2.9: Congestion window size of TCP with RED

2.5.2 Implementation Issues

Prior to presenting the simulation results, we describe a few implementation issues. In the simulations in this section, we use a sliding window of time-step size to estimate the

arrival rate from controlled and uncontrolled sources at the router. Time-step size is set to be 5 msec and the time-interval of the sliding window is chosen to be the round-trip time of the sources [46], i.e., the number of sliding window slots is equal to $rtt/0.005$. We estimate the arrival rate by computing the instantaneous arrival rate over the time-step size and averaging the sliding window length of current and previous instantaneous arrival rates.

Further, the actual number of flows is not needed at the router to compute the marking probability based on the equivalent rate based marking function. Computing the total controlled and uncontrolled rates (as opposed to the average arrival rate) is sufficient as the equivalent marking function is automatically "normalized."

In the simulations, the buffer size (also called queue limit) is set to be sufficiently large such that only marking functionality affects the transmission rate of controlled flows, i.e., physical dropping of the arriving packets does not occur due to queue overflow.

In practice, however, packet drops could occur due to finite buffers at routers. The equivalent rate based model in this chapter can be extended to such a finite queue length system by adding an equivalent rate based *dropping function* (thus the intermediate router has a pair of probability (p_m, p_d), where p_m and p_d are marking and dropping probability computed based on the equivalent rate based model), since the dropping function for a finite size of queue in a queue based system is a step function. Further, this model can be extended to more complicated queue based dropping function (e.g., RED) than a step function. However, in this chapter, we restrict to an intermediate router with only marking functionality.

2.5.3 Experiment 1: Proportional Fair Controller

Figures 2.4 and 2.5 show the average (over flows) instantaneous throughput as well as the long-term average over the entire simulation time (straight horizontal line in the figures) for the REM and RED controllers. As probabilistic marking is employed at the router (with both queue based and equivalent rate based marking), the starting times of flows and transmission pattern of uncontrolled flows are randomized, the instantaneous rates will not be identical in a path-wise sense. However, a good match between the two schemes implies that *the statistical behavior should be close to each other.*

Both figures include the results for different network parameters such as the round-trip time, the load of uncontrolled flows, and the burstiness of uncontrolled flows. Through these results, we can compare the long-term behavior of a queue based and a rate based scheme. The results show that there is about less then 5% difference between the rate based and the queue based scheme. Further, the instantaneous rate show similar statistical path behavior.

In this simulation, the arrival rate could exceed the service capacity. In this situation, while the equivalent rate based marking marks all the packet, not all packets are marked in the practical packet systems. In addition to the Poisson approximation, we posit that one of the reasons for performance difference between the rate based model and the queue based model is due to such a fact.

However, if the system scale is large enough, this effect becomes small for the following reason: when the arrival rate exceeds the capacity, the queue-length increases, leading to an unstable queue. Thus, as the system scale increases (indexed by n), the queue length is of order n, and thus an increasingly large fraction of the incoming packets will be marked (as the marking function is unscaled). This is supported by the simulation results that even 100 controlled and uncontrolled flows are enough to decrease the performance error to less than 5%.

2.5.4 Experiment 2: TCP Controller

Figures 2.6 and 2.7 plot the throughput with the REM and RED controllers respectively, and Figures 2.8 and 2.9 plot the congestion windows (averaged over flows) as a time series. Further, Figure 2.10 shows that complementary distribution function of the congestion window for a typical TCP flow, for two different loads and with the RED controller at the router. The plots indicate that the window sizes are statistically very similar for the queue based marking and the equivalent rate based marking. Thus, this indicates that replacing a queue based marking function by its equivalent rate based function does not statistically affect the end host's behavior.

2.5.5 Experiment 3: Sensitivity to Change of Network Connections

In this simulation, we study the sensitivity of the equivalent rate based marking function to the change of network connections. The simulation starts with 100 uncontrolled and 200 controlled sources. After 100 seconds, the number of controlled sources is reduced to 100. Then from 200 seconds, the number of uncontrolled sources increases to 150. The bottle-neck bandwidth maintained at 100×100 pkts/sec and the mean transmission rate of an uncontrolled source is sustained, which means that the uncontrolled load after 200 sec is changed to 75%. In Figure 2.11, we can see the robust behavior of the proposed rate based marking function even with the change of network connections.

Appendix

2.5.6 Proof of Lemma 2.4.1

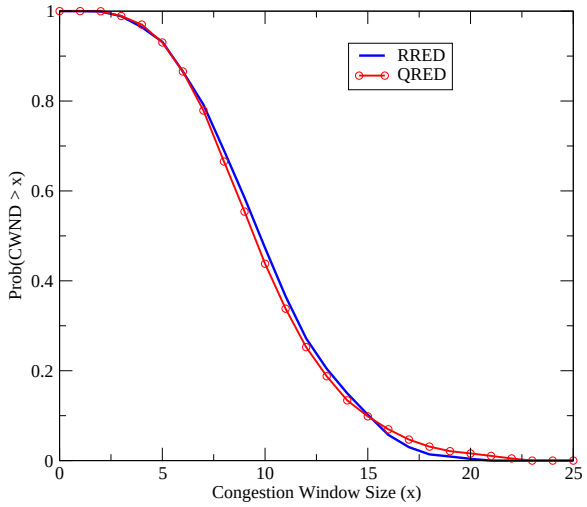
Proof. Let $J \triangleq J(a_n)$ be the number of jumps (number of arrivals) of the trajectory $a_n(t),\ t \in [0,T]$. It can be shown that for n large enough, $J < \infty$ almost surely, since $a_n(\cdot)$ converges to $a(\cdot)$, a finite rate Poisson process, in the Skorohod topology (thus, for n large enough, and over a compact time interval, the number of arrivals in $a_n(\cdot)$ is the same as the number of arrivals in $a(\cdot)$; see also the proof of Lemma 2.4.2). Let $\{t_n^j,\ ,\ t_n^j \le T,\ j = 1,2,\ldots,J\}$ be the jump times, respectively.

Next, let $\xi_n(r) = y_n(r) - x(0)r$. Then, we have

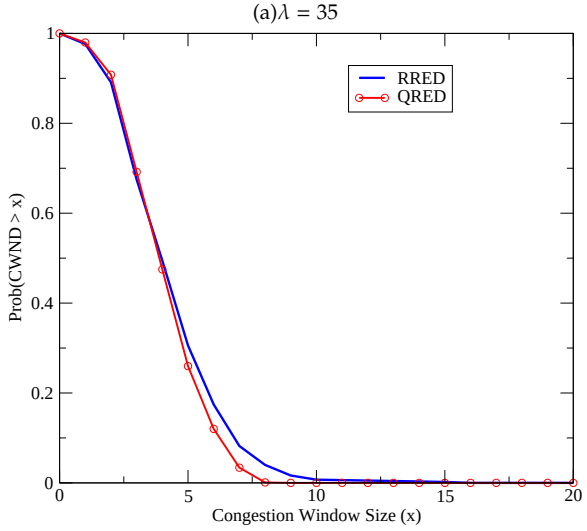
$$\begin{aligned} |\xi_n(t) - \xi_n(r)| &= \left| n \int_{r/n}^{t/n} (x_n(z) - x_n(0))\, dz \right| \\ &\le \left| n \int_{r/n}^{t/n} Mz\, dz \right| \\ &= \frac{nM}{2}\left(\left(\frac{t}{n}\right)^2 - \left(\frac{r}{n}\right)^2\right) = \frac{M}{n}(t^2 - r^2) \end{aligned} \tag{2.27}$$

Then, from the definition of $q_n(t)$ and $\tilde{q}_n(t)$,

$$\begin{aligned} q_n(t) &= \sup_{r\in[0,t]} [a_n(t) - a_n(r) + x(0)(t-r) + \xi_n(t) - \xi_n(r) - c(t-r) + q_n(r)] \\ \tilde{q}_n(t) &= \sup_{r\in[0,t]} [a_n(t) - a_n(r) + (t-r)x(0) - c(t-r) + \tilde{q}_n(r)] \end{aligned}$$



(a)λ = 35



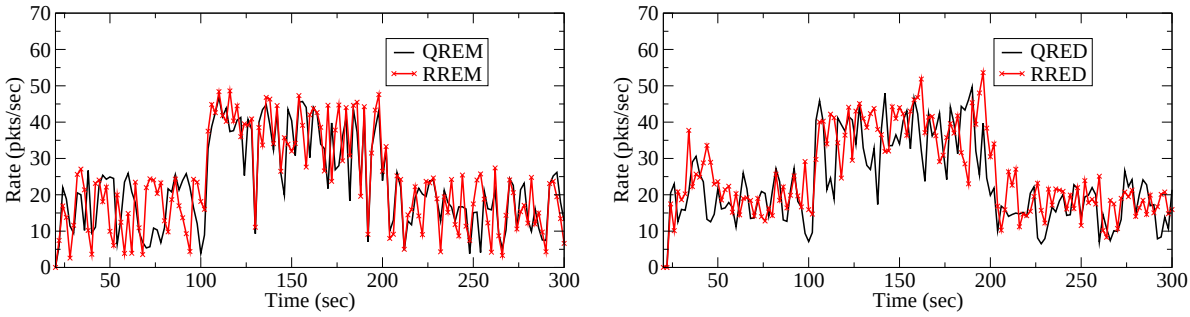
(b) λ = 50

Figure 2.10: Congestion window size distribution of a typical TCP source with RED: ON-OFF(0.1) and rtt = 200 msec

Let $\Delta\tilde{q}_n(s) = |q_n(s) - \tilde{q}_n(s)|$, $s \in [0, T]$. Then, it suffices to show that for given $\epsilon > 0$, there exists N such that $\forall n > N$, we have

$$\sup_{s\in[0,T]} \Delta\tilde{q}_n(s) \quad < \quad \epsilon,$$

since convergence with respect to uniform topology implies convergence with respect to Skorohod topology.



(a) Proportional Fair, REM (b) TCP, RED

Figure 2.11: Sensitivity to Change of Network Connections

Consider any two jump times t_n^j and t_n^{j+1}. Then, we have

$$\begin{aligned}\Delta\tilde{q}_n(t_n^{j+1}) &\leq \Delta\tilde{q}_n(t_n^j) + |\xi_n(t_n^{j+1}) - \xi_n(t_n^j)| \\ &\leq \Delta\tilde{q}_n(t_n^j) + \frac{M}{n}(t_n^{j+1} + t_n^j)(t_n^{j+1} - t_n^j) \\ &\leq \Delta\tilde{q}_n(t_n^j) + \frac{M}{n}(2T)(t_n^{j+1} - t_n^j)\end{aligned}$$

Further, for any $z_n^j \in [t_n^j, t_n^{j+1}]$,

$$\begin{aligned}\Delta\tilde{q}_n(z_n^j) &\leq \Delta\tilde{q}_n(t_n^j) + \frac{M}{n}(2T)(z_n^j - t_n^j) \\ &\leq \tilde{q}_n(t_n^j) + \frac{M}{n}(2T)(t_n^{j+1} - t_n^j) \qquad (2.28)\end{aligned}$$

Thus, it is enough to check $\Delta\tilde{q}_n(s)$ when $s \in \{t_n^j, , t_n^j \leq T, j = 1, 2, \ldots, J\}$. Noting that $\Delta\tilde{q}_n(0) = 0$ since $\tilde{q}_n(0) = q_n(0)$, we have

$$\begin{aligned}\Delta\tilde{q}_n(t_n^1) &\leq \frac{M}{n}2T(t_n^1 - 0) \\ \Delta\tilde{q}_n(t_n^2) &\leq \frac{M}{n}2T(t_n^1 - 0) + \frac{M}{n}2T(t_n^2 - t_n^1) \\ &= \frac{M}{n}2T(t_n^2) \\ &\ldots\end{aligned}$$

By induction, we have

$$\Delta\tilde{q}_n(t_n^j) \leq \frac{M}{n}2T(t_n^j) \leq \frac{M}{n}2T^2 \qquad (2.29)$$

From (2.28) and (2.29), the result follows. □

2.5.7 Proof of Lemma 2.4.2

Proof. By definition of convergence in Skorohod topology, for a given $0 < \delta < 1$ and sufficient large n, we can find a strictly increasing, continuous function λ_n of $[0, T]$ onto itself such that

$$\begin{aligned} \sup_{s\in[0,T]} |a_n(\lambda_n(s)) - a(s)| &< \delta \\ \sup_{s\in[0,T]} |\lambda_n(s) - s| &< \delta, \end{aligned} \tag{2.30}$$

and for n large enough, we have

$$J \triangleq \mathcal{J}(a) = \mathcal{J}(a_n), \tag{2.31}$$

where $\mathcal{J}(\cdot)$ is the number of jumps over the space $\mathcal{D}([0, T] : \mathcal{R}^+)$. The fact that $a(s), s \in [0, T]$ is a Poisson process with a finite rate ensures that we have a finite number of jumps almost surely over the finite interval of time. In addition, as any "extra jump" would lead to a distance of 1 which contradicts condition (2.30), Equation (2.31) follows. Let us denote the arrival times of a and a_n by $\{t^j, j = 1, \ldots, J\}$ and $\{t_n^j, j = 1, \ldots, J\}$, respectively. We know that the arrival times of $a(t)$ ($a_n(t)$) are equivalent to the jump times of $q(t)$ ($q_n(t)$). Also, we know that

$$\sup_{1\le i\le J} |t_n^i - t^i| < \delta,$$

for sufficient large n from (2.30). For a function $\lambda_n(s)$ satisfying (2.30), we choose a piecewise linear function such that $\lambda_n(t^i) = t_n^i, i = 1, \ldots, J$. This construction implies that $\lambda_n(s)$ is a continuous and strictly increasing function over the interval $[0, T]$. To complete the proof, it suffices to show that $\sup_{s\in[0,T]} |q_n(\lambda_n(s)) - q(s)|$ is arbitrarily small. Let

$$\Delta q_n(s) = |q_n(\lambda_n(s)) - q(s)|$$

Then, we have the following recurrence relation

$$\Delta q_n(t^{j+1}) \quad \leq \quad \Delta q_n(t^j) + \delta(c - x(0)),$$

since $q_n(\lambda_n(s))$ and $q(s)$ has only one jump at time t^{j+1} in the interval $(t^j, t^{j+1}]$. Thus, the queue size difference is only that due to the amount of service with rate $c - x(0)$ over the time difference $|\lambda_n(s) - s|$. Further, for any $r \in [t^j, t^{j+1}]$, we have

$$\Delta q_n(r) \quad = \quad \max[\Delta q_n(t^j), \Delta q_n(t^{j+1})],$$

as the $q_n(\cdot)$ and $q(\cdot)$ are piece-wise linear between jumps. Thus, it is enough to check only the jump times of the process $q(s), s \in [0, T]$. Thus, defining $t^0 = 0$, and choosing n large enough such that $|q_n(0) - q(0)| < \delta$, we have

$$\begin{aligned} \sup_{s\in[0,T]} |q_n(\lambda_n(s)) - q(s)| &\leq \max_{0\leq j\leq J} \Delta q_n(t^j) \\ &\leq \delta \max\{1, J(c - x(0))\} \end{aligned}$$

Choosing δ small enough, the result follows. □

Chapter 3

FluNet: A Hybrid Internet Simulation/Emulation Environment for Fast Queue Regimes

3.1 Overview

Motivated by the scale and complexity of simulating large-scale networks, recent research has focused hybrid fluid/packet simulators, where fluid models are combined with packet models in order to reduce simulation complexity as well as to track dynamics of end-sources accurately. However, these simulators still need to track the queuing dynamics of network routers, which generate considerable simulation time-complexity in a large-scale network model.

In this chapter, we propose a hybrid simulator – FluNet – where queueing dynamics are not tracked. The FluNet simulator is predicated on a fast-queueing regime at bottleneck routers, where the queue length fluctuates on a time-scale that is much faster than the time-scale of end systems. FluNet does not track queue lengths at routers, but instead, uses an equivalent rate based model at router queues, and queue-based AQM schemes (such as RED) are replaced by such an equivalent rate-based model. This allows us to simulate large-scale systems, where the simulation "time step-size" is governed only by the time-

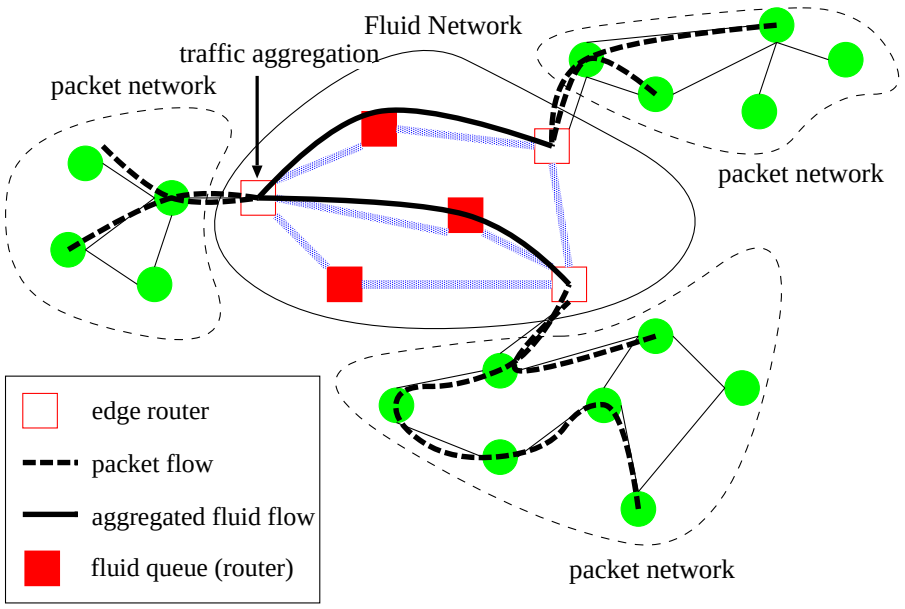


Figure 3.1: Hybrid Simulation Framework

scale of the end-systems, and not by that of the intermediate routers; whereas a fluid model based simulator that *tracks* queue-length would require decreasingly smaller step-sizes as the system scale size (such as the number of flows and link capacity) increases. We validate our model using both an *ns-2* and Linux based implementation. Our results indicate a good match between packet systems and the associated FluNet system.

3.2 Introduction

The Internet has experienced tremendous growth in both scale and speed, and the control and management of the Internet is becoming an ever more important issue. To model and understand the behavior of such networks, several widely-used discrete event-driven simulators are available (such as ns-2, GlomoSim, QualNet, PDNS, and SSFNet, see their references in [47]) in the area of simulation. However, event-driven simulation of large scale network systems with a significant number of users and flows is difficult due to simulation time complexity.

Recently, there have been significant efforts on developing (approximate) fluid model

based simulators to address the time complexity of discrete event simulators. These simulators can be classified into *pure fluid model based approach*, and *hybrid model based approach.* Pure fluid model based approach includes [12–15], where the authors are primarily interested in rate based fluid modeling of TCP sources, AQM algorithms, and their interactions. On the other hand, hybrid model based approach [16–20] integrate packet models along with fluid models to enable hybrid simulation. Hybrid simulators have both advantages of (a) accurately tracking source dynamics (as the sources in the simulator are typically modeled using packet networks), and (b) reducing simulation time complexity by fluid approximation in the core network (the system scale permits a fluid approximation to be accurate [9]) and by "dimensional collapse" due to traffic aggregations inside the core network (see Figure 3.1).

Existing hybrid simulators, such as in [16], integrate fluid models with packet systems by measuring data rate from packet flows over short time-intervals, and use these rate measurements to drive a fluid simulator. Thus, the fluid simulator is approximated by a discrete-time simulator, where the time-step corresponds to the time interval used for rate measurement in the packet system. In other words, the continuous time fluid process is approximated by a discrete time sampled system, with the sampling frequency chosen to be large enough to accurately represent the fluid model queue length dynamics. The resulting discrete time fluid simulator consists of a collection of fluid queues (which evolve in discrete time), whose dynamics in-turn, are used to evolve the packet system (for example, fluid queue lengths are used to determine packet marking probabilities).

An important consideration in such a hybrid simulator is the selection of the discrete time-step. This time-step interval should be chosen small enough such that (fluid) queue lengths can be accurately tracked. However, an overly small time-step will lead to a large simulation time-complexity, as this complexity linearly increases with the sampling frequency.

A potential problem with this approach is that the rate of queue length variations increases as the system scale size (i.e., the number of flows and the link capacity) increases. Thus, increasingly large sampling frequencies are required as the system scale size increases. To understand this intuitively, consider an M/D/1 queue accessed by n flows,

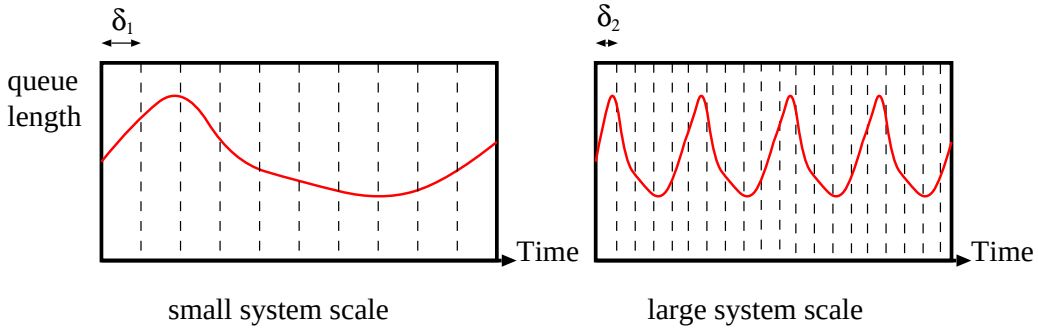


Figure 3.2: Queue length trajectories with different system scales

where each flow is modeled by a Poisson arrival process with parameter λ, and the system capacity is nc. Then, from standard queueing theory, the (mean) busy period (i.e., the time interval over which an empty router buffer fills up and empties again) is expressed by $\frac{1}{n(c-\lambda)}$. Note that this time step is *inversely proportional* to n, the system scale size. *This implies that we need progressively smaller step-sizes (increasingly higher sampling frequency) to capture the fluid queue dynamics accurately as the system scale (i.e., n) increases.* (To see this, observe that at least one sample is needed per busy period of the queue in order to track the queue length process). Thus, due to increasingly larger sampling frequencies with system scale, such a "queue-tracking based approach" is not scalable in terms of simulation time complexity.

This phenomenon is illustrated in Figure 3.2, where the queue length is sampled over a step size δ_1 in the left figure, but the step size needs to decrease to δ_2 in the right figure as the number of flows and the capacity of the router increases, in order to accurately capture the queueing dynamics.

To address this, we develop a hybrid network simulator – **FluNet** – which uses the statistical behavior of the intermediate routers, and results in a system, where *the step-size is independent of the system scale size, but only depends on the time-scale of end-system source dynamics.* The FluNet simulator is predicated on a "fast queue regime" at bottleneck routers, and does not track queue lengths at the core-routers, but instead, uses an *equivalent rate based marking/dropping model* (for a given queue based AQM model) that depends on the (stochastic) stationary behavior of the router queue.

The main assumptions of the fluid model in FluNet are that *(i)* there is sufficient randomness generated by both end systems as well as intermediate routers (e.g., unresponsive flows such as multimedia and short-lived web-mice flows, probabilistic behavior of AQM algorithm, and TCP variability due to its window-based flow control), and *(ii)* the system scale size is large. In this situation, the queue dynamics at the intermediate routers occur on a much *faster* time-scale than that of the end system controller. *Thus, it is reasonable to expect that queueing dynamics are not visible to the end system controller. Instead, the queueing behavior at the router affects the end system controller only through the "statistical behavior of the queue."* This allows us to simulate large-scale systems, where the simulation "step size" is governed only by the time-scale of the end-systems, and not the time-scale of queue dynamics at the intermediate routers. Note that the time-scale of the end-systems is governed by the window dynamics of the congestion controller (e.g., TCP), which is, in turn, determined by the round-trip time. This is fixed for each flow and does not significantly change with the system scale (n), resulting in a fixed step-size that does not scale with n. Whereas a fluid simulator that *tracks* the queue length would require decreasingly smaller step sizes (and hence, increasing time-complexity) as the system scale size increases for an accurate simulation.

Another advantage of FluNet is that it facilitates the implementation of hybrid simulator of large scale systems with real-time end-to-end flows (i.e., real-time emulation), where the actual (real-time) data flows are injected to the core network, which is implemented by a simulation hardware. Clearly, in real-time emulation, *time* complexity is not a key issue, but our main concern is *implementation* complexity. Due to the limited processing power of the simulation hardware to handle clock timer, implementing very small step-sizes in a queue tracking based fluid simulator may not be possible (or at least difficult and costly). However, since the step-size is not scaled with the system scale in FluNet, it would be easier to implement FluNet with the same setup of the simulation hardware. This motivates us to implement and test FluNet in a real operating system (e.g., Linux) (see Section 3.7).

Our main contributions are the following:

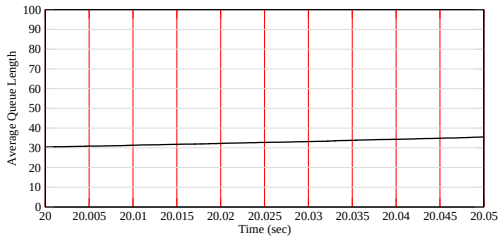
(i) We develop a fluid model using the equivalent rate based approach for a queue based algorithm, and study the practical issues in using the developed model for simulation.

Our model is predicated on the fast queue regime, which is reasonable to study for large-scale systems with sufficient network randomness. The main advantage of our model over conventional queue based fluid models is that it is governed only by the time-scale of the end-systems, and not the intermediate routers, resulting in the simulation step size that does not need to decrease with increasing system scale. In other words, for a fixed step-size and parameters, our model becomes progressively better as the scale of the system increases.

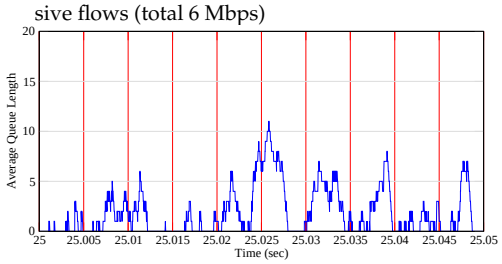
(ii) By implementing FluNet in a popular discrete event simulation (ns-FluNet) and in a real Linux operating system (real-FluNet), we validate our model and its feasibility for both simulation as well as real-time emulation with real traffic. The simulation/measurement results show a good match between a packet system and the associated FluNet system under various network topologies and traffic conditions.

In this chapter, we assume that the entire simulated/emulated system including TCP and intermediate routers is ECN (Explicit Congestion Notification) [48]-enabled, such that TCP sources respond to the ECN congestion signal from receivers, and AQM algorithm in the intermediate routers functions with the "marking mode" in order to notify sources of the incipient congestion by setting ECN bit of traversing packets. However, our study is not limited to ECN-enabled systems, and FluNet can be easily extended to systems with non-ECN mode as well as simple DropTail queues (see Section 3.4.5).

We comment that another interesting approach is presented in [49], where the authors describe a procedure where a sampled version of the traffic is fed to scaled-down version of the network model, and then linearly extrapolate the results from the scaled-down system to the original large-scale system. Essentially, the authors argue that in several ways, a slowed down system mimics the larger system. However, the authors in [49] point out that correlated events (e.g., burst of packet losses, for instance) breaks the linear-scaling hypothesis, and causes the scaled simulation to deviate from the real system. Recent work in [46] applies network calculus based on the mathematical theory of Min-Plus (or Max-Plus) algebra to fluid modeling of network dynamics. We also remark that results in [46] suggest that fluid queue based simulation could perform poorly when the

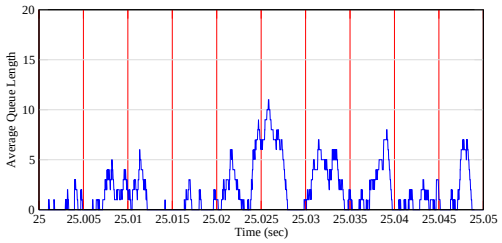


(a) 20 Mbps bottleneck bw, 20 TCP, and 20 unresponsive flows (total 6 Mbps)

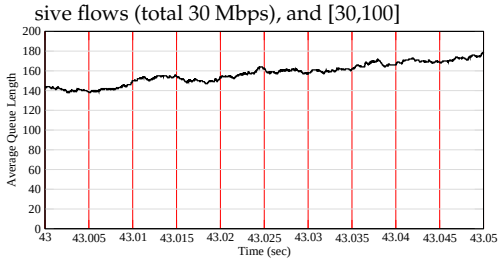


(b) 100 Mbps bottleneck bw, 100 TCP and 100 unresponsive flows (total 30 Mbps)

Figure 3.3: Rate of queue variations for different system scale sizes



(a) 100 Mbps bottleneck bw, 100 TCP, 100 unresponsive flows (total 30 Mbps), and [30,100]



(b) 100 Mbps bottleneck bw, 100 TCP, 100 unresponsive flows (total 30 Mbps), and [300,1000]

Figure 3.4: Rate of queue variations for different queue threshold parameters: [a,b] = [`min_th` , `max_th`] of RED

bottleneck buffers are not saturated. This can be understood from the fact that unsaturated buffers correspond to a system with fast queueing dynamics, where tracking queue length trajectories (i.e, a fluid queue based approach) may not be feasible. More related work on the alternate models for fast queue regimes are discussed in Section 3.4.3.

3.3 Fast and Slow Queue Regime

In this section, we present the *ns-2* based simulation results to show that the rate of queue variations indeed become faster for larger system scales. Recall that FluNet is designed to operate at a fast queue regime, and thus these simulation results provide a practical motivation for the FluNet. The simulations are performed in a single bottleneck network (see Figure 3.7(a)) with RED algorithm [21], accessed by TCP and unresponsive ON-OFF sources. The end-to-end source-destination round trip time is set to be 200 msec. In the simulation results, shown in Figures 3.3 and 3.4, we plot the queue length trajectory at the bottleneck link over a time interval of 50 msec.

First, in Figure 3.3, we plot the queue length trajectories for different system scale sizes, where the system size in Figure 3.3(b) is scaled by a factor of 5. We observe that increasing system scale size generates faster queue dynamics, as previously discussed in Section 3.2.

Second, we note that fast queue variations can also occur if the mean queue length is kept small, and thus resulting in many "busy cycles" over a short interval of time. For example, with the RED algorithm, the parameters, `min_th` and `max_th`, determine the queue length thresholds at which packet marking/dropping occurs (see Figure 3.4.5 for details). In a Drop-Tail queue, the physical queue size limit corresponds to this threshold parameter. By choosing these parameters to be small, we can ensure that packets are marked or dropped aggressively, resulting in "small queues." This in turn leads to fast queue length variations as illustrated in Figure 3.4(a). On th other hand, setting these parameters to be large results in slow queue length variations as shown in Figure 3.4(b).

Experimental motivation and justification for fast queue regimes is also based on [50,51], where a fast queueing regime corresponds to a small queue regime in a large-scale system. Note that the term 'small queue' corresponds to the queue length when normalized with the system capacity. Such a regime seems reasonable for large-scale systems based on arguments presented in [50,51]. It has been argued that the required buffer size does not have to scale linearly with the system scale size [50], and in fact can be fixed, independent of the system scale size [51]. This implies that in large systems, the buffer fluctuations will be fast, because the buffer size normalized to the link capacity shrinks. Thus, the queue length *normalized* with the capacity will be small, leading to fast queue dynamics. More discussions on buffer size scaling will be provided in Section 3.4.3.

3.4 Fluid Model of FluNet

3.4.1 Basic Model and Intuition

As discussed in Section 3.2, the key idea of FluNet is to find an equivalent rate based model for a given queue based AQM algorithm (e.g., RED [21]), and the main assumptions of finding the equivalent rate based model are sufficient randomness and large scale size in the Internet. Indeed, there is a sufficient amount of randomness in the Internet mainly due to

unresponsive flows, flow initiation and terminations, and probabilistic marking/dropping mechanism implemented in the routers. Recent studies [45, 52] show that unresponsive sources contribute to about 70% - 80% of the Internet flow counts[1]. Typical examples of such unresponsive flows include multimedia (video and audio) flows and web mice (short-lived HTTP flows). Further, the scale of the current Internet is very large, e.g., in the Sprint backbone, OC-192 links have been installed in several POPs (Points-of-Presence) with a total of approximately 11 M TCP connections being captured during one hour in those back-bone routers [53].

Under such a regime, we will have a considerable number of "cycles" in the queue dynamics of the intermediate routers even over a small interval of time, where one "cycle" corresponds to the time interval over which an empty router buffer fills up and empties again (technically, the regeneration time). In particular, the queue dynamics occur on a much *faster* time-scale than that of the end system controller [4, 32]. In order to understand this intuitively, consider a router of capacity $n \times c$ accessed by n TCP flows and n unresponsive flows. Then, the time scale of a TCP source rate update is the order of $1/c$ (since its rate update is clocked by the ACK packets from the receiver), whereas the time scale of a router queue "cycle" is in the order of $1/(nc)$. Thus, it is reasonable to expect that queueing dynamics are not visible to the end system controller. Instead, the queueing behavior at the router affects the end system controller only through *the statistical behavior of the queue.*

In the previous chapter (as well as the authors in the related work [32, 55]), we have formalized the above heuristic, and derived a formal limit relating a queue based marking function and an equivalent rate based marking function as the number of flows n goes to infinity under different simple assumptions. However, these results do not discuss practical issues (e.g., queue averaging of AQM algorithms, TCP burstiness, and drop functionality at routers) in applying such limiting models to simulation/emulation. In this chapter, we extend the equivalent rate based model developed in the previous chapter to account for such practical issues. Further, along with the insight described earlier in this chapter that the step-size does not need to shrink if appropriate statistical behavior is used, we develop two implementations (one in ns-2, and another in Linux) of FluNet, and present simulation

[1]However, the volume of data in unresponsive flows contribute to about 10% - 20% of the total traffic volume of the Internet.

and measurement results.

3.4.2 Equivalent Rate Based Model Considering Queue Averaging

For a given queue based AQM model, we develop an equivalent rate based model considering queue averaging. The model of the n-th system consists of a single bottleneck router (with its capacity nc) fed by n TCP flows and n unresponsive flows. For notational simplicity, we have assumed an equal number of TCP and unresponsive sources. The results in this chapter hold even if they are not the same, as long as the ratio of the number of TCP flows and the number of unresponsive flows is finite.

We denote the fluid rates of individual TCP flows in the n-th system by $\{x_n^i(t), i = 1, \ldots, n\}$, where $x_n^i(t)$ denotes the arrival rate of a TCP flow i at time t. Let $x_n(t)$ be the average TCP arrival rate at time t, i.e., $x_n(t) = \frac{1}{n}\sum_{i=1}^{n} x_n^i(t)$. For the well-defined initial conditions, we assume that $x_n^i(0) \to x^i(0)$, and $x_n(0) \to x(0)$, as $n \to \infty$. We assume that $\dot{x}_n^i(\cdot)$ is bounded (i.e., the transmission rate is Lipschitz), and $x_n^i(\cdot)$ is bounded by some constant L. This in-turn implies that $x_n^i(\cdot)$ is Lipschitz continuous with some parameter $M < \infty$ [9]. Further, we assume that $p_m^q(\cdot)$ is also Lipschitz continuous.

We model each unresponsive flow by means of a point process $A_n^i(t)$, that represents the cumulative number of arriving packets from flow i until time t in the n-th system. We assume that each $A_n^i(t)$ has the same distribution as a simple stationary point process A that satisfies the following assumptions [30, 37]:

Assumption 3.4.1.

(i) There exists $\lambda > 0$ such that $E[A(t)] = \lambda t$ for $t \in [0, \infty)$.

(ii) There exists $\theta_0 > 0$ and $K < \infty$ such that

$$\lim_{t \to 0^+} E[e^{\theta_0 A(t)} \mathbf{1}_{A(t)>K}] \quad = \quad 0,$$

where $\mathbf{1}_B = 1$ if the event B is true, and 0 otherwise.

(iii) $\liminf_{t\to\infty} \frac{t\Lambda(x,t)}{\log t} > 0$, *where* $\Lambda(x,t) = \sup_{\theta \in \mathcal{R}}[\theta x - \frac{1}{t}\log E[e^{\theta A(t)}]]$.

Assumption 3.4.1 states that each unresponsive arrival process satisfies the properties that *(i)* multiple packets from a single unresponsive source do not arrive at the same time, *(ii)* all arriving packets are of the same size and *(iii)* the unresponsive arrival process has a finite intensity (see [30] for further details).

Let $A_n(t) = \sum_{i=1}^{n} A_n^i(t)$ be the *total cumulative number of arrivals* until time t due to *unresponsive flows*, and $X_n(t) = \sum_{i=1}^{n} x_n^i(t)$ be the *total arrival rate* at time t due to TCP flows. We define the total volume of arrivals (due to the TCP flows) until time t by $Y_n(t)$, i.e., $Y_n(t) = \int_0^t X_n(z)dz = n \int_0^t x_n(z)dz$.

Associated with the router is a queue based marking function (AQM algorithm), denoted by $p_m^q(\bar{Q}_n(t))$, where $\bar{Q}_n(t)$ is the weight-averaged queue length (i.e., an exponential weighted moving average is implemented where the queue length is averaged over an appropriate time-scale in order to absorb a certain level of burstiness of incoming traffic). *Note that our marking function is based on the actual queue lengths, not the scaled queue length.* For simplicity, we assume that the employed AQM algorithm has only marking functionality and the infinite physical queue limit. A similar argument holds for a practical AQM algorithm with a finite queue limit and dropping functionality, or a simple Drop-Tail algorithm (see Section 3.4.5).

For a fixed $T > 0$, and for large n, we are interested in studying the queue length process (which measures the volume of data at the router) over the time-interval $[0, \frac{T}{n}]$. Even over this small time interval, we will show that the queue reaches "steady-state" behavior. This occurs due to the fact that the capacity is very large (nc), and causes the queue to "regenerate" an arbitrarily large number of times over the interval $[0, \frac{T}{n}]$. However, from a single *end-system* (the user) point of view, this corresponds to a very short interval of time. Thus, one can expect that the end-user will only perceive the statistical "steady-state" queueing behavior. The results in this section quantify the above heuristic.

Let us denote the *instantaneous* queue-length process at the router by $Q_n(t)$ (the subscript n indicates that the system scale size is n), and the weight-averaged process by $\bar{Q}_n(t)$, which is given by $\bar{Q}_n(t) = w_n \bar{Q}_n(t - \delta_n) + (1 - w_n)Q_n(t)$, where $0 < w_n < 1$ is the queue-averaging parameter for n-th system and $\delta_n = 1/(nc)$, where δ_n corresponds to the time-scale of the queue variation at the router (see also [12]).

In [21], the authors provide a guideline on the choice of the parameter w_n. Essentially, the authors in [21] argue that w_n is chosen such that a fixed burst of packets (i.e., L back-to-back packets from a single flow) should be allowed into the router without this burst being marked. This burst tolerance is chosen to account for TCP window behavior and cumulative ACKs, which lead to a burst of packets being transmitted from a single TCP source, instead of the packets being spaced apart. However, observe that as the number of flows and capacity increases, the *normalized packet burst size* decreases (normalization with respect to link capacity).

In particular, consider a bottle-neck router with capacity nc, and fed by n independent arrivals each of which has a packet-burst of size L packets (i.e., L back-to-back packets from a single flow). Then, if the flows are independent, it is unlikely that the packet bursts from various flows will synchronize and form a single large burst of nL. This heuristic is supported by [56], where the authors show that when multiple flows are aggregated, and the individual flows have different burstiness but equal rates, the burstiness of the aggregate flow is determined by the burstiness of the individual constituent flow which has the maximum burstiness. In other words, as the number of flows and the bottle-neck link capacity increases, the burstiness of aggregate incoming flows remains constant. This implies that the queue averaging parameter w_n needs to become smaller as the system scale increases (because the normalized packet burst size decreases). Motivated by this argument, we make the following assumption:

Assumption 3.4.2. $w_n \overset{n\to\infty}{\longrightarrow} 0$.

For any $s \in [0, \frac{T}{n}]$, the *instantaneous* queue length process is given by:

$$\begin{aligned} Q_n(s) &= \sup_{r\in[0,s]} \left[A_n(s) - A_n(r) + Y_n(s) - Y_n(r) - nc(s-r) + Q_n(r)\right] \\ &= \sup_{r\in[0,s]} \left[A_n(s) - A_n(r) + n\int_r^s x_n(z)\,dz - nc(s-r) + Q_n(r)\right] \end{aligned}$$

Recall that the weighted averaged queue length process is given by $\bar{Q}_n(t) = w_n\bar{Q}_n(t-\delta_n) + (1-w_n)Q_n(t)$, where $0 < w_n < 1$ and $\delta_n = 1/nc$.

Next, we define a queue length process $q(t)$ as follows:

$$q(t) = \sup_{r\in[0,t]} [a(t) - a(r) - (c - x(0))(t - r) + q(r)], \tag{3.1}$$

where $a(t)$ is a Poisson process with parameter λ. Intuitively, $q(t)$ is the queue length process of M/D/1 queue with Poisson arrival rate λ and capacity $c-x(0)$. Then, we have the following result on the marked volume of data experienced by a typical i-th TCP flow.

Now, let us study the processes $(X_n, Y_n, A_n, Q_n, \bar{Q}_n)$ over a *slowed-down* time-scale, i.e., for $t \in [0, T]$, let $q_n(t) = Q_n\left(\frac{t}{n}\right)$, $\bar{q}_n(t) = \bar{Q}_n\left(\frac{t}{n}\right)$, $a_n(t) = A_n\left(\frac{t}{n}\right)$, and $y_n(t) = Y_n\left(\frac{t}{n}\right)$. Then, from the definition of $Q_n(t)$ and $\bar{Q}_n(t)$, we have for any $t \in [0, T]$,

$$\begin{aligned} q_n(t) &= \sup_{r\in[0,t]} [a_n(t) - a_n(r) + y_n(t) - y_n(r) - c(t - r) + q_n(r)] \\ \bar{q}_n(t) &= w_n\bar{q}_n(t - 1/c) + (1 - w_n)q_n(t) \end{aligned} \tag{3.2}$$

Theorem 3.4.1.

$$\frac{1}{T/n}\int_0^{\frac{T}{n}} x_n^i(y)p_m^q(\bar{Q}_n(y))\,dy \overset{n\to\infty}{\longrightarrow} x^i(0)\frac{1}{T}\int_0^T p_m^q(q(y))\,dy \tag{3.3}$$

Theorem 3.4.1 states that in the large n regime, the time-average volume of marks experienced by i-th TCP flow over the interval $[0, T/n]$ (LHS) can be well approximated by the marked volume at the M/D/1 queue with Poisson arrival rate λ and capacity $c - x(0)$ (RHS). *Note that the original unresponsive arrival process is not necessarily a Poisson process.* In other words, for n large enough, and fixed T, the interaction between the router queueing process and the congestion controller at a fixed user occurs only through $\frac{1}{T}\int_0^T p_m^q(q(y))\,dy$ (the marking probability in the limiting system).

Proof. We will show that

$$\bar{q}_n(t) \overset{w}{\to} q(t), \quad t \in [0, T] \quad \text{over} \quad \mathcal{D}([0, T] : \mathcal{R}^+), \tag{3.4}$$

where $\mathcal{D}([0, T] : \mathcal{R}^+)$ is the space of RCLL (Right Continuous with Left Limit) trajectories over the interval $[0, T]$. Then, from Theorem 3.2 in [54], and using the Lipschitz continuity

of $p_m^q(\cdot)$ and $x_n^i(t)$, the result follows. Thus, we focus on the proof of (3.4) in the rest of this section.

First, we will prove that for any fixed $\epsilon > 0$, we can find N_1 such that $\forall n > N_1$,

$$\|\bar{q}_n(t) - q_n(t)\| \quad < \quad \epsilon \quad \text{over} \quad \mathcal{D}([0,T] : \mathcal{R}^+), \tag{3.5}$$

where $\|\cdot\|$ is the Skorohod metric.

From (3.2), we have

$$\begin{aligned} \|\bar{q}_n(t) - q_n(t)\| &= \|w_n \bar{q}_n(t - \frac{1}{c}) + (1 - w_n)q_n(t) - q_n(t)\| \\ &= w_n \|\bar{q}_n(t - \frac{1}{c}) - q_n(t)\| \leq w_n \sup_{t \in [0,T]} |q_n(t)|. \end{aligned}$$

Then, (3.5) follows from that fact that $\sup_{t\in[0,T]} |q_n(t)|$ is almost surely finite, and from Assumption 3.4.2.

Next, from Theorem 3.2 in [54], we have

$$q_n(t) \quad \stackrel{w}{\rightarrow} \quad q(t) \quad \text{over} \quad \mathcal{D}([0,T] : \mathcal{R}^+).$$

Then, from the Skorohod representation theorem [38], we can find processes $q_n'(t)$ and $q'(t)$ in $\mathcal{D}([0,T] : \mathcal{R}^+)$ such that $q_n(t) \stackrel{\text{dist}}{=} q_n'(t)$, and $q(t) \stackrel{\text{dist}}{=} q'(t)$, where $\stackrel{\text{dist}}{=}$ means "equivalence in distribution." Then, for the same ϵ in (3.5), we can find N_2 such that $\forall n > N_2$,

$$\|q_n'(t) - q'(t)\| \; < \; \epsilon \quad \text{over} \quad \mathcal{D}([0,T] : \mathcal{R}^+). \tag{3.6}$$

Let $\bar{q}_n'$ be the weighted averaged process of q_n'. By the triangle inequality of Skorohod norm, for the same ϵ in (3.5) and (3.6), $\forall n > N = max(N_1, N_2)$, we have

$$\|\bar{q}_n' - q'(t)\| < \|\bar{q}_n' - q_n'(t)\| + \|q_n'(t) - q'(t)\| < \epsilon + \epsilon = 2\epsilon.$$

Since ϵ is arbitrary, this completes the proof. □

Thus, the system with aggregate (over flows) unresponsive rate of $n\lambda$, and with aggre-

gate (over flows) TCP rate of nx can be represented by M/D/1 system of fixed service rate of $c-x$, and an arrival process that is Poisson with parameter λ (even if the actual system does not have Poisson arrivals). Further, we observe that $q(t)$ is a regenerative process when $\frac{\lambda}{c-x}$, and $x < c$, and from the ergodic theorem for a regenerative process, for large enough T, the following definition is a good approximation of the marked volume of data at the router. In this context, we define an equivalent rate based marking function $p_m^r(x)$ as follows:

$$p_m^r(x) = \begin{cases} E_{\pi_\lambda^{c-x}}[p_m^q(Q)] & \text{if } \frac{\lambda}{c-x} < 1 \text{ and } x < c, \\ 1 & \text{if } x \geq c \text{ or } \frac{\lambda}{c-x} \geq 1, \end{cases} \tag{3.7}$$

where Q is the stationary queue length random variable and π_λ^{c-x} is the stationary distribution of an M/D/1 queue with capacity $c - x$ and arrival rate λ.

3.4.3 On the Alternate Models and Discussion

In this section, we compare the model that we use in the FluNet with other alternate models. These alternate models are derived based on the different assumptions on (i) buffer size scaling and (ii) randomness in the network.

The buffer size scaling has been classified into three regimes [55]: (a) *small buffer regime* ($B = \Theta(1)$, i.e., independent of the system scale size n), (b) *intermediate buffer regime* ($B = \Theta(n^\alpha)$, $0 < \alpha < 1$), and (c) *large buffer regime* ($B = \Theta(n)$), where n is the system scale size (i.e., the number of flows and the system capacity), and B is the buffer size at the bottleneck router.

Traditionally, network design (at the back-bone routers) has been predicated on the large buffer regime, i.e., the buffer is provisioned in linearly proportional to the system scale size [57]. However, recent experimental studies on the buffer size scaling show that we can achieve sufficient statistical multiplexing gain and high network utilization in small buffer regime [51] as well as intermediate buffer regime [50]. Further, analytical results [30,31] show that arbitrary small loss probability and high throughput are guaranteed even with small buffer regime. The small buffer regime is based on the intuition that a large number of flows multiplexed at the large capacity router and *randomness* (which leads to

de-synchronization among flows) in the network enables sufficient statistical multiplexing without large buffer size.

The major sources of randomness in the network are generated by (i) unresponsive flows (ii) TCP flows. The randomness in unresponsive flows is due to real-time multimedia flows, short-lived web-mice, and random connection arrival and departure to the network. On the other hand, the randomness in TCP flows is due to inability to precisely model a window-based flow control, initial slow-start phase, and probabilistic AQM algorithms in the intermediate routers. Different models can be derived, depending on the different assumptions on the randomness by these two types of flows, their relative magnitude and time-scale of variability.

The authors in [32, 55] assume that the time-scales of randomness by both TCP and unresponsive flows are in the same order, thus aggregate TCP and unresponsive flows form a Poisson process with parameter $(\lambda + x)$, where λ and x are the mean arrival rates of unresponsive and TCP flows, respectively. Thus, the limiting system is approximated by an M/D/1 system with capacity c and Poisson input with parameter $\lambda + x$. On the other hand, our preliminary work [54] assumes that the randomness of TCP flows happens at much slower time-scale than that of unresponsive flows, i.e., over a small interval of time, the mean arrival rate of TCP flows looks constant. This assumption leads to a M/D/1 system, but with capacity $c - x$ and Poisson input with parameter λ.

The assumptions and analytical models in [32, 55] can be viewed as a "conservative" interpretation of the Internet, i.e., even *controlled* TCP flows has the same magnitude of randomness as *random* unresponsive flows. However, the models in [54] analyze the Internet on a "optimistic" assumption, i.e., the randomness TCP flows over a short time-interval is ignored (i.e., the inter-packet jitter by TCP flows are negligible in the large system scale size), and assume that the randomness by unresponsive flows dominant in the network. Intuitively, in the system with the randomness of both TCP and unresponsive flows, more marking/dropping of packets occurs than the system with randomness only with unresponsive flows, at the intermediate routers. This intuitive interpretation can be justified by the standard queueing theory. Let $\rho_1 = \frac{\lambda}{c-x}$, and $\rho_2 = \frac{\lambda+x}{c}$. Then, for all values

of $\lambda > 0$ and $x > 0$, such that $\lambda + x < c$, we have $\rho_1 < \rho_2$[2]. Thus, from P-K formula of M/D/1, the system with ρ_2 marks/drops (on an average) more packets at the routers than that with ρ_1, leading to TCP throughput difference in both systems.

FluNet can be implemented either the model in [32,55] or that in this paper. Both models in the FluNet framework are easily implementable, by looking up the marking/dropping probability in the table (see 3.4.6 for details). However, for space limitation, we have provided simulation results using the model in this chapter, as the result due to both models are very close with 100 flows.

3.4.4 Choice of Measurement Interval and Simulation Step Size

Implementing a simulator based on (3.7) requires measurement of x (TCP arrival rate) and λ (unresponsive arrival rate). In our implementation, this is done by choosing a small measurement interval $\bar{M}$, which satisfies *(i)* that *the arrival rate from TCP flows does not vary significantly over* $\bar{M}$, and *(ii)* that *there exists a sufficient number of regenerative cycles in the router queue over* $\bar{M}$, enabling us to see the statistical stationary behavior (i.e., the Poisson approximation holds for the chosen value of $\bar{M}$). In our experiments, we have chosen one (minimum over end-to-end flows) round-trip time for $\bar{M}$, which is independent of the system scale size, n. This choice seems to be reasonable, since the transmission rate of a TCP flow is clocked by the received ACK feedbacks, and we can also see a significant queue variations over one round-trip time (a large number of regenerative cycles) in large scale systems (e.g., in the plots of Figures 3.3 and 3.4, even over $\frac{1}{4}$ of a round-trip time in the router with 100 flows, we can see more than 20 cycles of queue regeneration). Note that since we focus on the flows passing through a WAN, the minimum round trip-time over flows will be large enough to see the stationary behaviors at the core-routers.

Additionally, we evolve the dynamics of the end-systems multiple times (say, W times) per a measurement interval $\bar{M}$. If this "splitting" of a measurement interval is not done, ACKs will aggregate over a measurement interval at a source, and lead to spurious bursts of packets, resulting in incorrect end-system behaviors. Thus, we evolve the simulator with a time step-size of $\delta = \frac{\bar{M}}{W}$. Note that $\bar{M}$ and W do not depend on the system scale size,

[2]In transients, it is possible to have that $\lambda + x > c$, in which case ρ_1 can be greater than ρ_2.

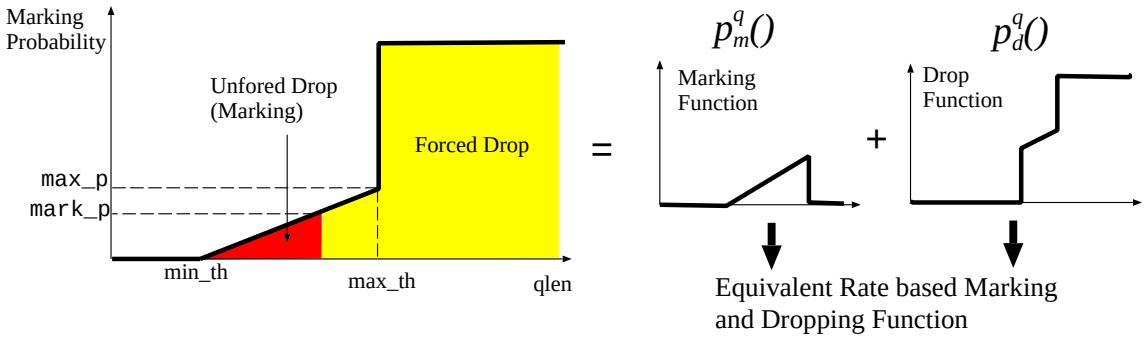


Figure 3.5: Marking and dropping in RED with marking mode

n. W depends only on the number of packets per flow in a round-trip time (i.e., congestion window size), so that the packets can be "clocked out" without spurious bursts[3]. Thus, as W and $\bar{M}$ are independent of n, δ is also independent of n. This is the key observation that leads to the reduced simulation time complexity for large scale systems.

3.4.5 Marking and Dropping in AQM

A realistic fluid model of a AQM algorithm should incorporate packet dropping as well. This is due to the fact that *(i)* a queue has a finite queue limit size, and *(ii)* some AQM algorithms drop the incoming packet when the current (averaged) queue length is larger than a threshold. For instance, the RED [21] algorithm marks packets below a queue threshold, but drops packets if the current (average) queue length exceeds this threshold (see unforced and forced drop region in Figure 3.5). Packets are also dropped if the packet buffer overflows.

This dropping functionality can be incorporated into the equivalent rate based fluid model simply by decomposing the queue based marking-dropping function into a pure drop and marking function ($p_m^q(\cdot)$ and $p_d^q(\cdot)$), as shown in Figure 3.5. Each of these two functions are then used in (3.7) to compute the equivalent rate based marking and dropping probability, denoted by $p_m^r(\cdot)$ and $p_d^r(\cdot)$, respectively. We comment that a simple Drop-Tail queue without marking functionality can also be implemented by setting its marking

[3]Note that this *does not* mean that "valid" bursts are not allowed. If a burst of ACKs arrive at a TCP source, the TCP source will in fact send a burst of packets by ACK-clocking.

function to be always zero and its drop function to be a step function (1 if qsize ≥ qlimit, and 0 otherwise).

3.4.6 Implementation of Rate based AQM

As discussed in Section 3.4.2 and 3.4.5, we have to compute $E[p_m^q(Q)]$ and $E[p_d^q(Q)]$ to determine the marking/dropping probability for a given fluid input rate. However, we may not always find a closed form expression to compute $E[f(Q)]$, where Q is the stationary queue length in an M/D/1 queue. Thus, our implementation uses numerically pre-computed values of the marking probability for each arrival rate (discretized suitably). This computation is performed using results in [42], where the authors have shown that the unfinished work U for a Poisson process with arrival rate λ, and with service rate μ has a steady state distribution of the form

$$\Pr(U > x) = 1 - (1 - \rho)e^{\rho x} H_{\lfloor x \rfloor}(x - \lfloor x \rfloor), \tag{3.8}$$

where $H_n(x)$ are polynomial functions (which can be computed recursively as shown in [42]), and $\rho = \lambda/\mu$. Then, by denoting the stationary workload of an M/D/1 queue with capacity c by V, we have $V = Q/c$, and $U = \mu V = (c - x)V$. Thus,

$$\Pr(Q > q) \quad = \quad \Pr\left(U > \frac{q(c-x)}{c}\right) \tag{3.9}$$

This is used to off-line pre-compute a table of marking/dropping probabilities for a large set of input traffic parameters (λ, μ), and AQM parameters, and use table lookups during run-time.

3.5 FluNet Architecture

3.5.1 Architecture Summary

FluNet consists of a hardware/software fluid-based core network (implemented in a Linux PC or within *ns-2*), and end systems that can either be implemented in hardware (real-FluNet) or within *ns-2* (ns-FluNet). The entire simulation framework consists of four

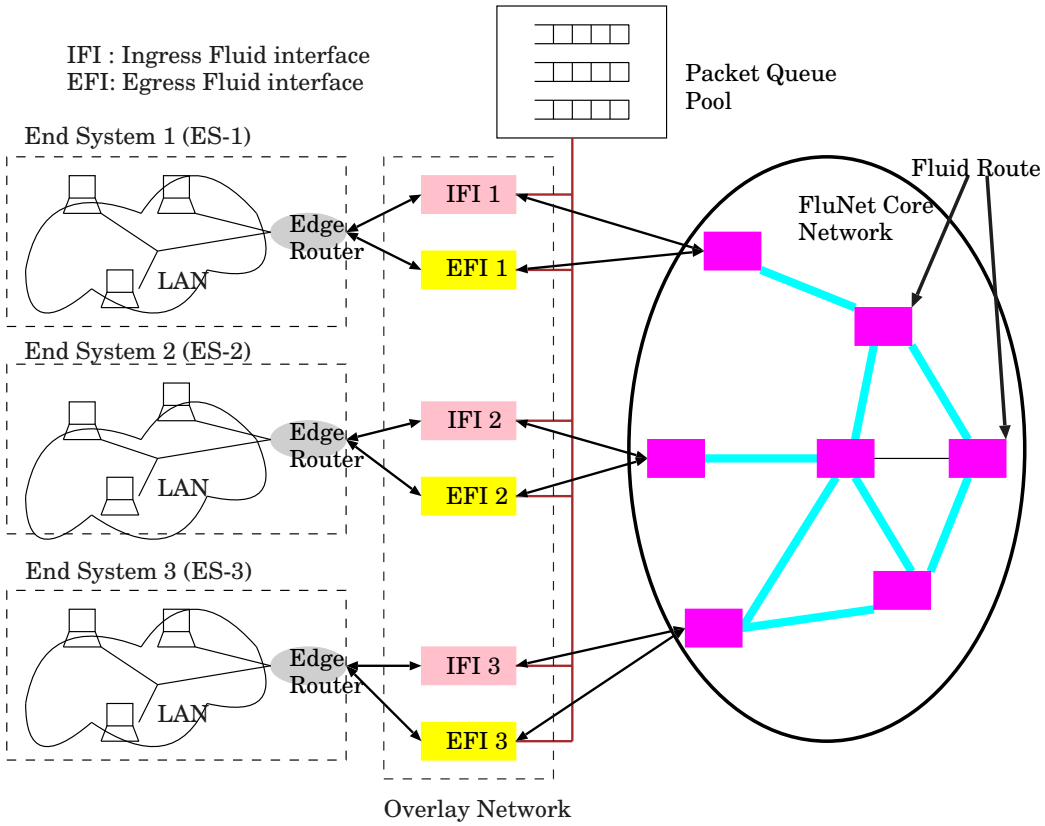


Figure 3.6: FluNet Architecture

components: *(i)* ingress fluid interface, *(ii)* fluid routers, *(iii)* egress fluid interface, and *(iv)* packet queue pool, as shown in Figure 3.6. Ingress and egress fluid interfaces reside at edge of the FluNet-core (a pair of ingress and egress fluid interfaces forms an interface node), and there are multiple fluid routers (depending on the topology of simulated network) inside the FluNet-core, where a fluid router corresponds to a packet router in the Internet-core.

To illustrate, let us consider the two packet streams between the end-systems of (ES-1↔ES-2) and (ES-1↔ES-3). The packets from ES-1 (destined to a node in ES-2 or ES-3) are first transmitted to the ingress fluid interface, IFI-1, which classifies them into two *classes* depending on the destination end-systems, ES-2 and ES-3. IFI-1 records per-class packet transmission *rate information* over successive time steps (i.e., δ), stores the received real packets at a queue (identified by a tuple (ingress fluid interface, egress fluid interface)) belonging to the packet queue pool, and forwards the rate information into the FluNet-

core. The forwarded rate information is processed by the FluNet-core, and the associated marked/dropped rate is computed at each fluid router using the equivalent rate based marking/dropping model in Section 3.4. The updated marked/dropped rate information is finally transferred to the egress fluid interfaces, EFI-2 and EFI-3, which marks/drops the real packets (fetched from the packet queue pool) physically based on the rate information computed at the fluid routers in the routing path, and forwards them to the destination networks in ES-2 and ES-3. We will describe the detailed procedure and issues in section 3.5.2.

The important thing that we note is that the real packets do not traverse the FluNet-core. The real packets are stored at the associated queue (corresponding to the ingress-egress pair) in the packet queue pool. Since only aggregate flow information between end-systems is transfered inside the FluNet-core, the maximum number of (aggregate) flows present within the FluNet-core is just $n_I(n_I - 1)$, where n_I is the number of the edge-routers of FluNet-core (e.g., $n_I = 3$ in Figure 3.6).

In this chapter, we focus on the data traffic injected only by end-packet systems, and the queue regimes at the intermediate routers at the FluNet-core. We can easily extend our analysis and implementation to more general scenario, where there exists data flows originated from and destined to the FluNet-core. The fluid representation of such flows inside the FluNet-core can be easily made by the techniques in [12, 14], and apply the fluid rates to the input of our rate based marking/dropping models.

3.5.2 Description of Components

A network *inside the FluNet-core* is modeled as a directed graph $G = (V, L)$, where V is a set of nodes (routers) and L is a set of links. Each link $l \in L$ has the propagation delay of γ_l and the capacity c_l. Let us denote $I \subset V$ to be a set of (ingress/egress) interface nodes, and $I = \{i_1, i_2, \ldots, i_M\}$, where M is the number of interface nodes inside the FluNet. Then, we define a class[4] k (which experiences the same route inside the FluNet-core), where $k \in I \times I \setminus \{(i, i) \mid i \in I\}$. Denote a set of links of the path of class k by $L_k = \{l_{k,1}, l_{k,2}, \ldots, l_{k,n_k}\}$,

[4]A class within the FluNet-core is a single fluid-flow that corresponds to multiple packet flows over the same path.

where $l_{k,n_i} \in L, i = 1, \ldots, n_k$, and n_k is the number of links in the path of class k.

Ingress Fluid Interface

At each ingress fluid interface, a step size δ is chosen (based on the criterion discussed in Section 3.4.4), and the total number of bytes that arrive from the end-system edge routers over this interval for each class is recorded over successive time step intervals, and this recorded rate information is transferred to the first link inside the FluNet-core in the routing path of the corresponding traffic class. The incoming real packets are stored in a packet queue (indexed by its class) of the packet queue pool.

At the end of each time step $s = 1, 2, \ldots$ (with each interval of δ), a vector $(\hat{t}_k[s], \hat{m}_k[s], \hat{d}_k[s], \hat{c}_k[s], \hat{u}_k[s])$ (which represent total, marked, dropped, TCP, and unresponsive received data volume, respectively) is generated corresponding to the aggregate data volume over the s-th time step for class k. Note that at the ingress fluid interface, the variable $\hat{d}_k[s]$ is initialized to be 0. Whether a packet is TCP/unresponsive or marked/unmarked is determined by checking the IP packet header CE and ECT field. At the end of each time step, the rate vector is transferred into the FluNet-core.

Fluid Router

The main task of a fluid router is to update the marked/dropped volume of data in the rate vector transferred from an ingress fluid interface or the previous fluid router in the routing path (by applying the rate based fluid model in Section 3.4), and to finally transfer those vectors to the associated egress fluid interface.

To understand the traffic interactions inside the FluNet, let us denote TCP, unresponsive, total, marked, and dropped received volume of arrival data corresponding to class k at link $l \in L$ over the s-th time step by $c_k^l[s]$, $u_k^l[s]$, $t_k^l[s]$, $m_k^l[s]$, and $d_k^l[s]$, respectively. Associated with every link $l \in L$ are (rate based) marking and dropping probability $p_m^l[s]$ and $p_d^l[s]$,

which are computed by (3.7) and its similar rate based dropping function[5]. Then, we have

$$c_k^{l_{k,i}}[s] = \begin{cases} \hat{c}_k[s], & i = 1, \\ \left(1 - p_d^{l_{k,i-1}}[\tilde{s}]\right) c_k^{l_{k,i-1}}[\tilde{s}], & i = 2, \ldots, n_k, \end{cases} \tag{3.10}$$

$$m_k^{l_{k,i}}[s] = \begin{cases} \hat{m}_k[s], & i = 1, \\ \left(t_k^{l_{k,i-1}}[\tilde{s}] - m_k^{l_{k,i-1}}[\tilde{s}]\right) p_m^{l_{k,i-1}}[\tilde{s}] + m_k^{l_{k,i-1}}[\tilde{s}], & i = 2, \ldots, n_k, \end{cases} \tag{3.11}$$

where $\tilde{s} = s - \tilde{\gamma}_{l_{k,i-1}}$, and $\tilde{\gamma}_l$ corresponds to the link l propagation delay computed in time-steps, i.e., $\tilde{\gamma}_l = \lceil \gamma_l / \delta \rceil$. Note that mean queueing delays can be easily incorporated by adding it to the link propagation delays. However, in sufficiently large scale systems, queueing delay is negligible, compared to link propagation delay. This is also justified by [50], where the authors show that buffers need to scale only as $\sqrt{n}$, whereas the capacity scales with n, which implies that the queueing delay is $O(\frac{1}{\sqrt{n}})$, while the propagation delay is $\Theta(1)$. Analogous equations to (3.10) and (3.11) hold for the unresponsive arrival rate ($u_k^{l_{k,i}}[s]$) and the drop rate ($d_k^{l_{k,i}}[s]$), respectively (except that $\hat{d}_k[s] = 0$, as discussed above). Finally, the total arrival rate of each class k is computed by the sum of TCP and unresponsive arrival rate, i.e.,

$$t_k^{l_{k,i}}[s] = c_k^{l_{k,i}}[s] + u_k^{l_{k,i}}[s] \tag{3.12}$$

Notice that update of $c_k^l[s]$ and $u_k^l[s]$ due to the dropping probability is needed, since it affects the TCP and unresponsive arrival rate at *the next link* (and thus, affects the computation of marking/dropping probability of next link).

As discussed in Section 3.4.4, in computing $p_m^l[s]$ and $p_d^l[s]$, we need the TCP and unresponsive rate information over $\bar{M}$ interval as their input. In our implementation, we use the TCP/unresponsive volume of data averaged over past W time steps, i.e., $\bar{c}_k^{l_{k,i}}[s] = \sum_{j=1}^{W} \frac{c_k[s-j+1]}{W}$, and similarly, $\bar{u}_k^{l_{k,i}}[s]$.

[5] We eliminate the superscript 'r' in (3.7) for notational simplicity.

Egress Fluid Interface

At the egress of the FluNet-core is the egress fluid interface connected to the destination end-system. When a rate vector, corresponding to a particular class k leaves the last fluid router (in the routing path of class k), this vector enters the egress fluid interface of the corresponding destination end-system after the propagation delay of link l_{k,n_k}. At each time-step s, the egress fluid interface $i \in I$ maintains a collection of vectors of *byte-counters* (not necessarily integers) for every class $k \in \{i\} \times I \setminus \{i\}$, where byte-counters consist of: *(i)* total counter ($T_k[s]$), *(ii)* mark counter ($M_k[s]$), and *(iii)* drop counter ($D_k[s]$). These byte counters, T_k, M_k, and D_k keep track of the amount of total data, the amount of marks, and the amount of drops received for the class k at each time step. Note that the egress fluid interface does not need the information of TCP or unresponsive data rate, since they are used only in computing marking/dropping probability inside the FluNet-core. At the end of each time-step (when a vector, (t_k^l, m_k^l, d_k^l) is transferred to the egress fluid interface), the interface increments three counters by the incoming quantities.

At each time-step s, if $T_k[s]$ is larger than or equal to the size of a packet in the packet queue (of, say, a class k) in the packet queue pool, the head-of-line (HOL) packet from the packet queue is transferred to the corresponding *destination egress fluid interface* by means of an *overlay network* (see Figure 3.6). Then, the egress fluid interface determines how to handle this real packet: drop, forward with marking, or forward without marking. First, the fetched real packet is dropped with probability $D_k[s]/T_k[s]$. Next, if the packet is not dropped, the value of the mark counter is now checked, and the ECN bit of the real packet is set to the value '1' with a probability $M_k[s]/T_k[s]$. This is simply an implementation of probabilistic marking corresponding to the fluid mark/drop rate. This packet is now forwarded to the edge-router of the destination end-system, which will suitably forward the packet to its final destination.

Once the packet has been forwarded or dropped, the byte counters are updated by the following: $M_k[s] = M_k[s](1 - sz/T_k[s])$, $D_k[s] = D_k[s](1 - sz/T_k[s])$, and $T_k[s] = T_k[s] - sz$, where sz is the size of the packet that is fetched and processed. The same procedure is repeated unless $T_k[s]$ is smaller than the HOL packet in the associated per-class packet queue.

3.6 NS-2 Simulation Results

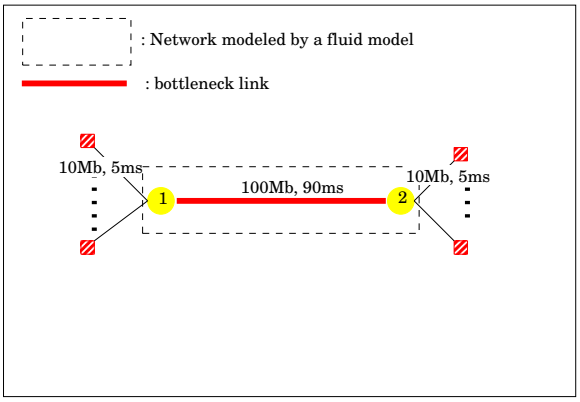
In order to validate FluNet, we have implemented FluNet in *ns-2* (ns-FluNet). Since [16] has a good reference framework for hybrid fluid/packet simulation, we adopted and modified the *ns-2* implementation in [16] by replacing its core fluid model implementation with our rate based model, while slightly modifying their interface between the fluid network and the packet network.

Our objective here is to compare a fluid queue based hybrid simulation (QFM) in [16] with FluNet, and determine the regimes for which each is suitable. Our base-line for comparison is a "packet" system, where no fluid approximation/models are used. We do not intend to validate the simulation time speed-up of FluNet over the associated packet system, since it has been already addressed in [16].

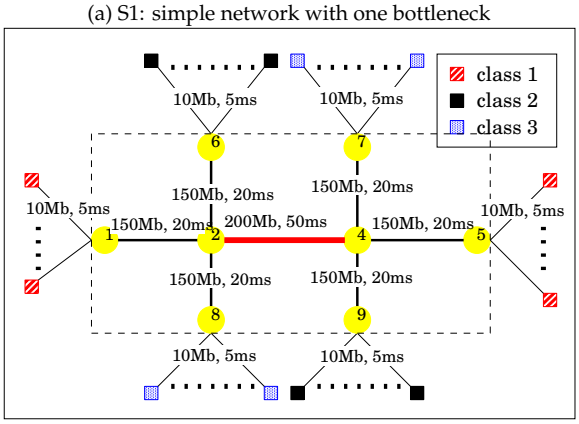
3.6.1 Simulation Environment

In this section, we have considered three different topologies, corresponding to a simple network with a single bottleneck link, and a large network with a single and two bottleneck links (denoted by S1, L1, and L2, respectively), as shown in Figure 3.7. The round-trip time of end-to-end flows in all topologies are set to be 200 msec, except for Experiment 3. The RED [21] is used as the AQM algorithm at the routers, where the parameters in RED are set to: `w_q` = 0.002, `gentle_` = false, `max_p` = 0.02, `mark_p` = 1.0, and `adaptive_` = false. With our parameter settings, `min_th` and `max_th` corresponds to the (average) queue length when the RED algorithm initiates marking and marking switches to dropping, respectively. As suggested in [58], in our simulations, we use TCP Sack as the end-system elastic data flows, and ON-OFF process as the unresponsive flows, respectively. In the ON-OFF process, the burst size of ON and OFF periods are exponentially distributed with mean 50 msec. The packet size for both TCP and the unresponsive flows are set to be 1000 bytes.

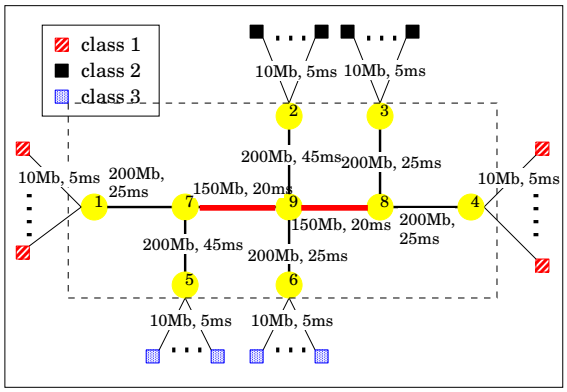
According to the selection rule of measurement interval and step size, we use $\bar{M} = 200$ msec, and $W = 40$, resulting in the step-size being set to 5 msec (i.e., $\delta = \frac{200}{40}$). The value of $W = 40$ is sufficient to avoid spurious bursts of transmitted TCP packets, since the steady-state congestion window size less than 25 packets, as shown in Figure 3.10.



(a) S1: simple network with one bottleneck



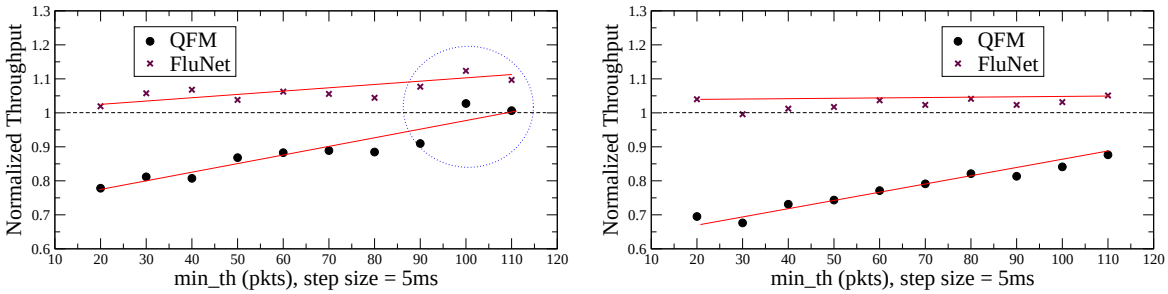
(b) L1: large network with one bottleneck



(c) L2: large network with two bottlenecks

Figure 3.7: Simulation Network Topology

In the simulation results, we present three statistics to investigate both steady-state and transient behavior, compared with the associated pure packet system: (i) normalized average throughput with respect to the packet system, (ii) congestion window sizes averaged over flows, and (iii) CWCR (Congestion Window Cut Ratio). CWCR represents



(a) 200 Mbps bottleneck bw, 100 TCP and 100 unresponsive (total 30 Mbps)

(b) 500 Mbps bottleneck bw, 250 TCP and 250 unresponsive (total 75 Mbps)

Figure 3.8: Normalized throughput of QFM and FluNet with different system scales as the queue threshold in RED changes. `max_th` =3×`min_th` .

the number of window cut events divided by the number of total transmitted packets, where window cut events include triple duplicate ACKs, retransmission timeouts, and ECN responses.

3.6.2 Experiment 1: FluNet and QFM under fast and slow queue regimes

In Figure 3.8, we plot the normalized average throughput of FluNet and QFM, where S1 network topology is used in the experiment. As discussed in Section 3.3, a *fast queue regime* results when the RED queue threshold parameters are small. We observe from Figure 3.8(a) that when the queue parameters are small, the throughput measured from FluNet is close to that of a packet simulation; whereas when the RED parameters are large, QFM outperforms FluNet, and the throughput measured with QFM is close to that measured with a packet simulation (see the dotted circular region in Figure 3.8(a)). This agrees with our intuition that FluNet will have better performance in a fast queue regime, but QFM outperforms FluNet in a slow queue regime.

However, for the same step-size and RED parameters, by scaling the number of flows and the bottleneck capacity, FluNet will again provide good results even with large RED parameters. This is because the RED parameters *when normalized by the capacity* again leads to a fast queue regime. *In other words, for a fixed step-size and parameters, our model will become progressively better as the scale of the system increases.* This is illustrated in Figure 3.8(b), where the system size (i.e., number of flows and capacity) is scaled by a factor of 2.5. Thus, we believe that *both QFM and FluNet are complementary,* and hybrid simulators should

Table 3.1: Average CWCR values over flows for large network topologies. (a,b,c,d) = (`min_th` , `max_th` , num of unresp flows, vol of unresp flows).

Name	Parameters	L1 ($\times 10^{-3}$)			L2 ($\times 10^{-3}$)		
		Pkt	FlN	QFM	Pkt	FlN	QFM
P1	(10,100,30,30Mb)	4.83	4.69	8.18	4.94	4.93	18.7
P2	(30,100,30,30Mb)	4.84	4.67	9.22	4.81	5.11	25.9
P3	(10,50, 30,30Mb)	5.06	4.95	9.05	5.32	4.95	17.9
P4	(10,150,30,30Mb)	4.67	4.54	7.72	4.69	4.45	7.22
P5	(30,100,10,30Mb)	4.91	4.66	9.34	5.28	4.74	28.1
P6	(30,100,90,30Mb)	5.15	4.90	9.76	5.22	4.93	26.6
P7	(30,100,30,10Mb)	2.45	2.29	6.36	2.72	2.62	4.63
P8	(30,100,30,50Mb)	14.4	14.4	15.2	12.7	11.3	17.8

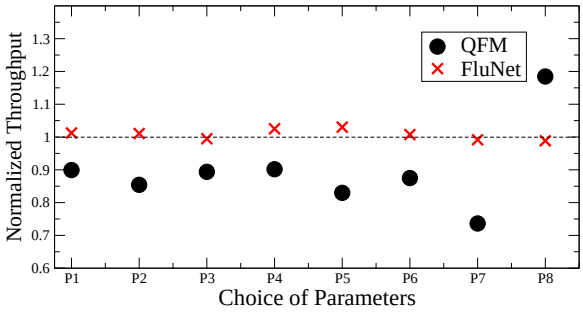
incorporate both these approaches, depending on the system scale.

3.6.3 Experiment 2: Larger network topologies

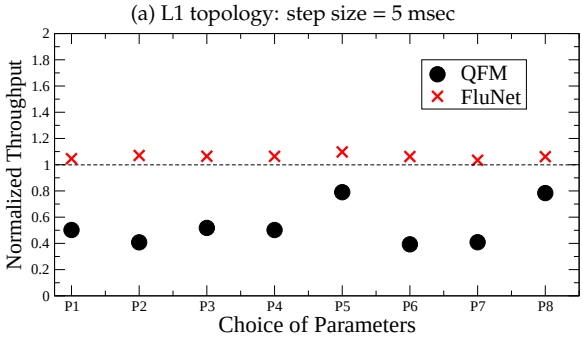
In this experiment, we present simulation results with four varying parameters: `min_th` , `max_th` , number of unresponsive flows, and total volume of unresponsive flows in the L1 and L2 network topologies, where there are three (TCP, unresponsive) traffic classes depending on the source-destination access network pair. Each choice of these parameters are denoted as P1-P8 (see Table 3.1), and the number of unresponsive flows and total volume of unresponsive flows in Table 3.1 correspond to the parameters in each class (not total network traffic flows). Further, for each traffic class, we run 50 TCP sources. We show only the performance results of class 1 TCP sources due to space limitation. We comment that similar results are obtained for different classes due to symmetry of three traffic classes.

Figure 3.9 and Table 3.1 summarize the simulation results of normalized throughput and CWCR Further, Figure 3.10 shows the associated CWND traces (averaged over flows) for selected experiments. As the experiments in Section 3.3 indicates, the parameter choices of queue thresholds and step-size in Figure 3.9 and Table 3.1 induce fast queue regimes. The results of FluNet match those of a pure packet network within an error of 5% for all cases considered. On the other hand, QFM has up to 45% throughput error, compared to the packet simulation.

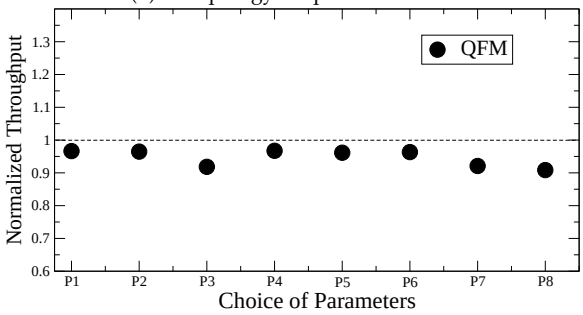
However, as shown in Figure 3.9(c), by decreasing the step-size to 1 msec and increasing



(a) L1 topology: step size = 5 msec

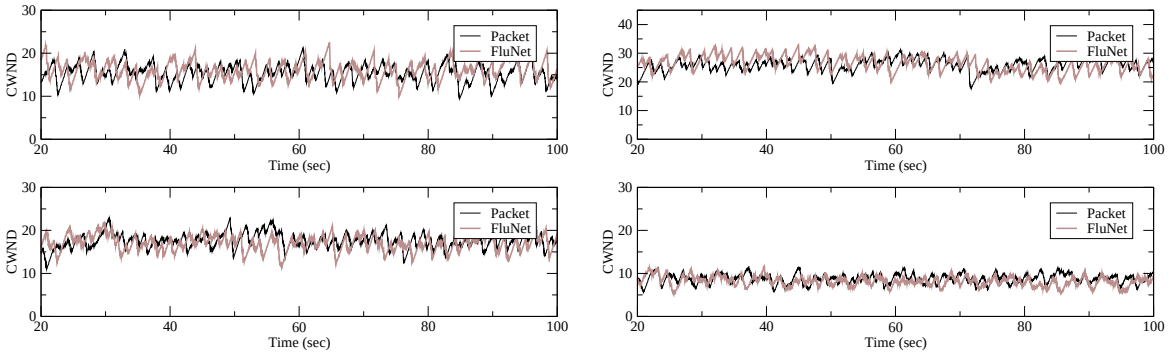


(b) L2 topology: step size = 5 msec



(c) L1 topology with 10 × queue thresholds as those in (a): step size = 1 msec

Figure 3.9: Normalized average throughput of QFM and FluNet



(a) # of UFs (10/90)

(b) vol of UFs (10/50) Mb

Figure 3.10: Average CWND traces in L1 network topology: `min_th` = 30, `max_th` = 100.

the queue thresholds (ten times as those in (a)), the results of QFM have a good match with those of packet systems. This is because with a step-size of 1 msec and with the large mean queue size, queue fluctuations are not severe, enabling QFM to accurately track queue dynamics.

FluNet is based on asymptotic models for router queues, where the number of flows are large. Indeed, FluNet simulation results match packet simulations closely when there are a large number of flows. For example, in a system with 4,200 flows, and the L1 network topology, the difference in average throughput between the packet system and FluNet is less than 4%. Importantly, in this section, we have shown that FluNet performs well even in a moderately scaled system, with only a few hundred flows. We skip the details for the large system (with many thousand flows) due to space constraints.

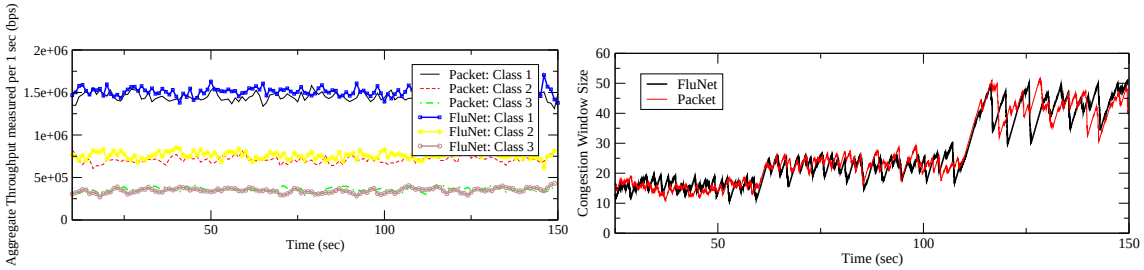


Figure 3.11: On the left: Avg throughput of TCPs with different round-trip times. On the right: Avg CWND traces with dynamic flow configuration

3.6.4 Experiment 3: Connections with different round-trip times, and dynamic scenarios

In this experiment, we first compare the performance results of FluNet with the associated packet network consisting of TCP sources with different round-trip times in the S1 network topology. TCP sources are divided into three classes (50 TCP session in each class) having different access propagation delays, such that there are three classes of TCP sources with round-trip propagation delays of 50 msec, 100 msec, and 200 msec, respectively. Figure 3.11(a) shows average instantaneous throughput (measured every 1 sec) for these three classes in both the packet and FluNet system. Again, we can observe that the packet system

and FluNet match closely.

Next, we measure the response of FluNet when flow configuration in the simulation changes dynamically. In the same configuration as above, all TCP sources of class 1 stop transmission at 60 sec, and then all TCP sources of class 3 stop transmission at 110 sec. Figure 3.11(b) shows the average congestion window traces of class 2. We observe that FluNet reacts to these changes in a similar manner to that in the packet network.

3.7 Linux Implementation and Experimental Results

3.7.1 Linux Implementation

First, we briefly describe the implementation of FluNet in a Linux operating system (real-FluNet). Though our implementation platform is Linux, we believe that it could be easily extended to other operating systems. We have implemented the FluNet-core using multiple processes in the user-level space at one computer, as shown in Figure 3.12. Network interface cards (Fast Ethernet cards) have been used to connect external end-systems with the FluNet-core. Associated with each network interface card are an ingress and an egress interface process (e.g., four ingress/egress processes in Figure 3.12). There is one FluNet-core process, which is responsible for simulating the fluid network. We have installed the packet queue pool in a distributed manner. In other words, individual packet queues (identified by a tuple (ingress,egress)) are maintained at the associated egress fluid interface process. Thus, for our example where we have four network interface cards, an egress interface process is responsible for maintaining four real packet queues (indexed by the corresponding ingress interface process). An ingress interface process is connected to every other egress interface using interprocess communication based on UDP sockets. In addition, a FluNet core process is fully connected to all the ingress/egress interface processes. The standard packet capture library (libpcap) is used for capturing packets at each ingress fluid interface process. To forward the real packet at an egress interface process, we use "raw socket" functionality.

By implementing FluNet-core at the user-level space (i.e., application level packet routing), we could obtain more flexibility in testing various AQM schemes and setting

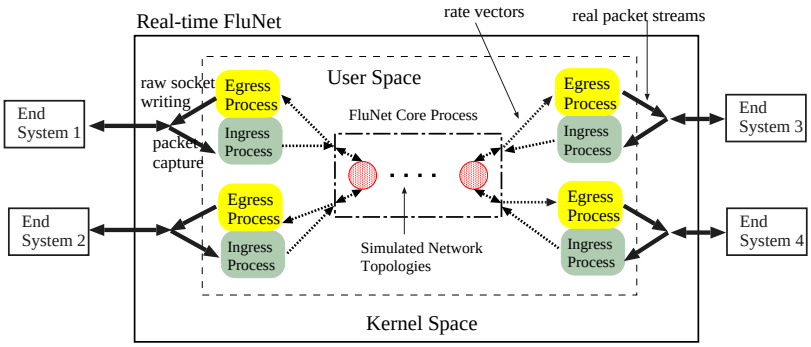


Figure 3.12: FluNet Linux implementation

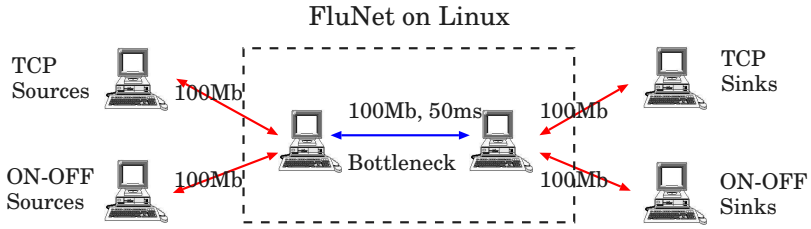
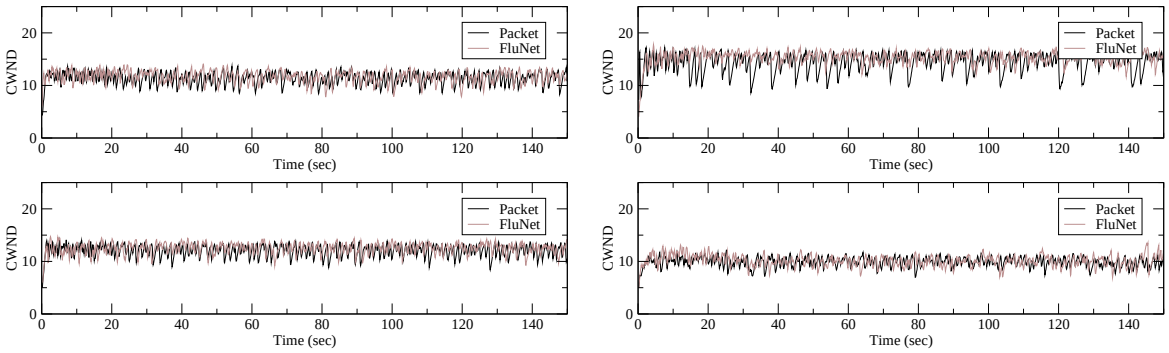


Figure 3.13: Network configuration with real-FluNet

their parameters. However, there is a trade-off between flexibility and the maximum speed of handling packets at the ingress/egress interface processes due to the cost of context switching between user and kernel space. From our test measurements, the maximum capture rate of standard libpcap is about 40-50 Mbps, which is not enough to test a Fast Ethernet interface. In our implementation, we use a modified version of standard libpcap (libpcap-mmap [59]), which increases the speed of capturing ability by using the mmap system call (i.e., memory-mapped I/O).

3.7.2 Experimental Results

In this section, we present measurement results of real-FluNet implementation in Linux. The network topology for measurement is shown in Figure 3.13. We consider a simple topology in this section so that we can implement an actual packet network with the identical topology and provide base-line measurements for comparison. Our real-FluNet implementation can be configured for other topologies are well, including those in L1 and



(a) # of UFs (30/70) (b) vol of UFs (10/40) Mb

Figure 3.14: Average CWND traces in real-FluNet, with RED parameters: `min_th`=30, `max_th`=100

L2 network topologies.

Two hosts are responsible for generating 50 TCP sources and a variable number of unresponsive ON-OFF sources. Two routers reside between source and destination pool, and all links are connected by Fast Ethernet 100 Mbps links (thus, the intermediate link between two routers is the bottleneck). In real-FluNet, both the bottleneck link as well as two routers are encapsulated into one real-FluNet computer. We use a 5 msec step size in real-FluNet. Both the TCP traffic as well as the ON-OFF traffic are generated using the iperf [60] traffic generator tool. Figure 3.14 and Table 3.2 provide the measurement results of real-FluNet in comparison with measurements with an identically configured packet network. The results show a good match between the two systems.

Table 3.2: Average throughput of real-FluNet

(# of UFs, vol of UFs)	Avg Th (P/F)	Error (%)
(30,30Mb)	1.210/1.179	2.5
(70,30Mb)	1.287/1.221	5.1
(40,10Mb)	1.550/1.496	3.5
(40,40Mb)	1.038/0.996	4.1

Chapter 4

On the Elasticity of Marking Functions: Scheduling, Stability, and Quality-of-Service in the Internet

4.1 Overview

Much of the research on Internet modeling and analysis has focused on the design of end controllers and network algorithms with the objective of stability and convergence of the transmission rate. However, the Internet is composed of a mixture of both (controlled) elastic flows and (uncontrolled) real-time flows. Uncontrolled real-time flows do not react to network congestion as well as they require a certain level of QoS guarantees. In this chapter, we study the effects of marking elasticity (which characterizes how aggressively the marking function responds to congestion) on the QoS for uncontrolled real-time flows, at a router accessed by both uncontrolled and controlled flows.

First, we derive lower and upper bounds on the queue overflow probability at a router of a single bottleneck system. Using this, we quantify the trade-off between stability for controlled flows and QoS guarantee for uncontrolled real-time flows as a function of marking elasticity. The results indicate that some marking functions may be "uniformly" better than others. In particular, among the marking functions that we have compared, our

bounds indicate that a rate based version of REM seems to provide the largest local-stability region for *any* given QoS requirement.

Next, we compare the capacity required at a router with only FIFO scheduling versus a router with priority scheduling (priority given to the real-time flows) for supporting a given QoS requirement (queue overflow probability). We quantify the "scheduling-gain" (in terms of supporting QoS for the real-time flows) of priority scheduling over FIFO scheduling, as a function of marking elasticity. We show that this scheduling gain decreases with more elastic marking functions. In other words, the difference in the required capacities with FIFO and priority scheduling for a fixed QoS (queue overflow probability) can be significantly reduced by increasing the marking elasticity.

4.2 Introduction

There has been extensive research on the modeling and analysis of the controlled elastic flows in the Internet by adopting differential equation based models of source controllers and AQM (Active Queue Management) algorithms. Much of this work has focused on the design of end host controllers and control algorithms (marking functions) at the intermediate routers for (global and local) stable end-to-end operation over the Internet by using control theoretic tools [1,3,6,7,10,24,27,61,62].

However, the Internet carries a mixture of traffic ranging from controlled non-real-time elastic data traffic to uncontrolled real-time traffic (e.g., voice and multimedia traffic). Uncontrolled real-time flows do not react to network feedback and requires tight QoS (Quality of Service) guarantees. From the perspective of network control and management, real-time flows are admitted into the network only if there are "sufficient" resources to satisfy their QoS requirements. On the other hand, non-real-time sources are always admitted into the network on a best-effort basis, i.e., real-time sources are given higher priority, and the remaining network resources unused by the real-time sources are allocated to the non-real-time sources.

One of the proposed architectures for providing differentiated QoS in the Internet is "Differentiated Service" model (DiffServ) [63], where users can belong to one of a small

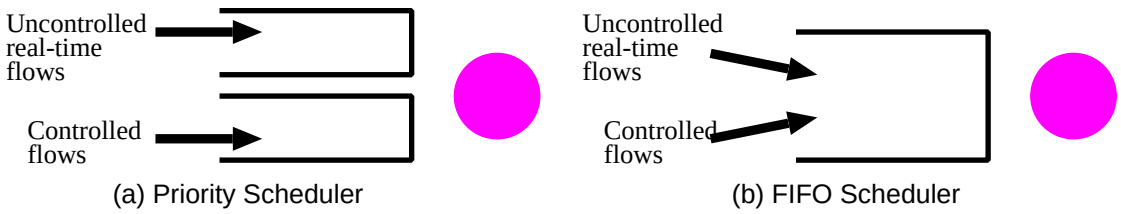


Figure 4.1: Priority and FIFO Scheduling Disciplines

number of classes, and QoS (such as delay, loss ratio, and throughput) for a user's data flow will be class-dependent. To implement such a service, routers in the Internet treat (schedule) packets from various classes in a differentiated manner depending on the class QoS specifications by adopting "priority" based scheduling algorithms (see Figure 4.1-(a) for a two class example).

In this chapter, we consider a network where resources are shared by uncontrolled real-time and controlled elastic flows, and packets in the router are scheduled in a first-come-first-serve manner (i.e., no differentiation) (see Figure 4.1-(b)). Over such a network, the behavior of uncontrolled real-time and controlled flows are coupled together, and the QoS experienced by uncontrolled real-time flows will be affected by the behavior of controlled flows (due to the flows sharing a common link). With this setup, it seems reasonable to believe that by appropriately designing an AQM mechanism (marking function) at intermediate routers, we can potentially provide the required QoS to the uncontrolled real-time flows *without any differentiation between real-time and non-real-time flows* at the routers.

The intuition is the following: an "aggressive" marking function will mark a larger number of controlled flow packets (for instance, those controlled by TCP) when a burst of packets (which causes congestion) arrive. This will cause the controlled flows to back-off, thus potentially decreasing the delay or packet loss probability experienced by real-time flows. In this chapter, we study the trade-off between packet marking [21] for controlled flows and the effect of this marking on the QoS of uncontrolled real-time flows.

We first characterize the "aggressiveness" of a marking function by its *elasticity*.

Definition 4.2.1. *Given any two marking functions $p_1(z)$ and $p_2(z)$, we say that $p_2(z)$ is* more

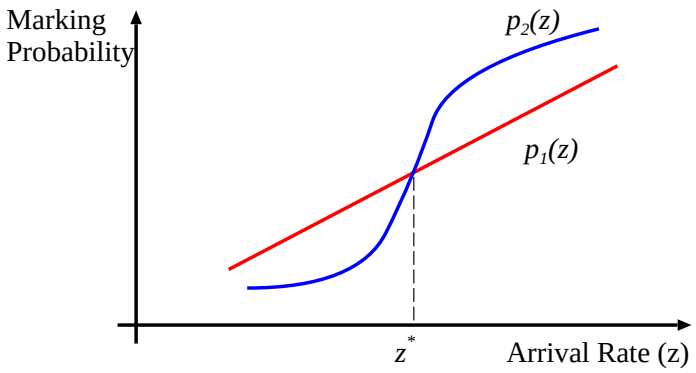


Figure 4.2: Elasticity of Marking Functions

elastic *than* $p_1(z)$ *if for any* $z \geq z^\star$, *we have*

$$p_1(z^\star) = p_2(z^\star), \qquad p_2(z) \geq p_1(z),$$

where $z^\star$ *is the equilibrium data rate at the router. Thus, the elasticity of a marking function corresponds to how aggressively the marking value changes as the arrival data rate exceeds the equilibrium rate (see Figure 4.2). However, note that the marking values at the equilibrium rate are equivalent with both marking functions, i.e.,* the link utilization at equilibrium is the same.

In this chapter, we study and quantify the following two trade-offs related with marking elasticity: stability-elasticity trade-off, and scheduling-elasticity trade-off.

Stability-elasticity trade-off: This refers to the trade-off between QoS-provisioning for uncontrolled real-time flows and stability for controlled flows. The key trade-off we quantitatively analyze is the following: the more elastic the marking function is, the better is the QoS experienced by uncontrolled real-time flows. However, this also leads to the negative property of less stability for controlled flows.

Scheduling-elasticity trade-off: This refers to the trade-off between the scheduling algorithm at the router and the elasticity of the marking function, with the performance metric being the QoS achieved for real-time flows, where the QoS requirement stipulates that the probability of packet (from uncontrolled real-time flows) loss should not exceed some threshold. This trade-off is quantified by means of the following: Given the requirement of (a) long-term throughput (equilibrium rate) of controlled flows and (b) QoS (queue

overflow probability)[1] for uncontrolled real-time flows, we compute the required link capacity at the router to support (a) and (b). To satisfy the long-term rate (for controlled flows) constraint (a), it is sufficient that the link capacity exceeds the sum of the mean rates of uncontrolled flows and the equilibrium rates of controlled flows (as long as the scheduling algorithm is work-conserving). However, the magnitude of the excess capacity depends on the given QoS constraint (b) for real-time flows, the marking function elasticity, as well as the scheduling algorithm. In particular, by giving absolute priority to the real-time flows (priority scheduling), this excess capacity can be minimized, as controlled flows do not affect the queue overflow probability for the uncontrolled flows. On the other hand, FIFO scheduling has the advantage of simple implementation (no per-class scheduling required), but at a cost of larger required link capacity. In this study, we quantify the "scheduling-gain" (i.e., the difference in the link capacity required with priority scheduling versus that with FIFO scheduling) of priority scheduling over FIFO scheduling, as a function of the marking elasticity.

The parameters that impact the source dynamics for a controlled flow are the round-trip delay, the elasticity of the marking function, and the rate of adaptation at the controlled source. In this chapter, we first model the dynamics of controlled flows by means of an instant adaptation algorithm, where the sources react to network feedback with no delay and adapt immediately to the equilibrium rate for a given network configuration. The instant adaptation scheme enables us to separate the effect of other parameters and to focus only on the elasticity of marking functions [7, 64]. In Section 4.5, we extend the discussion to a weighted proportional fair controller [1] with more complex temporal dynamics.

4.2.1 Main Contributions and Organization

The main contributions of this chapter are the following:

(i) Using the instant adaptation model for source dynamics, we derive lower and upper bounds of the queue overflow probability at a router, where a single buffer is shared by controlled and uncontrolled real-time flows. Using these bounds, we quantify the

[1] Throughput this chapter, the QoS requirement we consider is the queue overflow probability for the queue accessed by the uncontrolled real-time flows.

trade-off between stability for controlled flows and QoS-guarantee for uncontrolled real-time flows as a function of the elasticity of the marking function. The results indicate that some marking functions may be "uniformly" better than others. In particular, among the marking functions that we have compared, our bounds indicate that a rate based version of REM [10] seems to provide the largest local-stability region for *any* given QoS requirement, compared to other popular marking functions.

(ii) We next compare the capacity required at a router with only FIFO scheduling versus a router with priority scheduling for supporting a given QoS requirement. We quantify "scheduling-gain" of priority scheduling[2] over FIFO scheduling, as a function marking elasticity. We show that this scheduling gain decreases with more elastic marking functions. This indicates that by appropriately choosing the marking function and by using only a FIFO queue at the router, we can satisfy the QoS requirements of real-time flows without much over-provisioning.

(iii) We extend the results to the case with a weighted proportional fair controller at the source, and study the trade-off between stability and marking elasticity. Finally, we validate our analytical results using simulations.

The problem of determining the queue overflow probability has been studied extensively for queues [65–69] in the context of "open-loop" flows (i.e., there are no controlled flows). From a technical viewpoint, our research differs from the current literature in that we use a sample path large deviations framework to analyze a system, where the flows react to the link conditions via the congestion controller dynamics.

In the rest of this chapter, we begin with a description of the system model, parameterization of marking elasticity and the problem statement in Section 4.3. Next, in Section 4.4, we derive an upper and lower bound on the queue overflow probability, from which we analytically show stability-elasticity trade-off and scheduling-elasticity trade-off with the instant adaptation algorithm. In Section 4.5, we discuss the stability-elasticity trade-off with weighted proportional fair controllers. In Section 4.6, we provide numerical results

[2] Any class based scheduling policy will result in poorer QoS to real-time flows than strict priority queueing. Thus, a priority queueing system provides an upper bound on the QoS achieved by real-time flows with any class-based scheduling policy employed at the router.

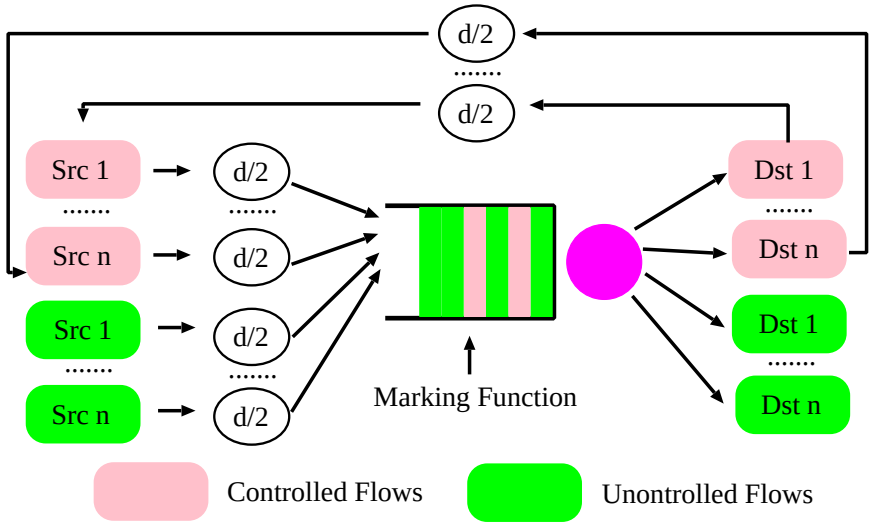


Figure 4.3: System Model

and simulation to validate our analysis.

4.3 System Model and Problem Statement

4.3.1 System Model

Consider the system shown in Figure 4.3. We consider a *single* discrete time queue with two types of flows: controlled flows and uncontrolled flows. We use the terminology *controlled flows* to refer to flows of data traffic which react and adapt their transmission rates to feedback from the network. An example of such a flow is a TCP (Transmission Control Protocol) flow. *Uncontrolled flows*[3] refer to data flows that do not react to network feedback. Examples of such flows include real-time video/audio, which usually require guarantee of real-time data transfer. The queue is fed by n independent identically distributed (over flows) stationary, ergodic uncontrolled flows and by n controlled flows (determined by a congestion control algorithm described later). The buffer size is scaled with n, and the link capacity of the corresponding queue is suitably scaled with n so that the queue is stable. Thus, the n-th system has a buffer of size nB, and a capacity of nC. For queue stability, we assume that $x^\star + y^\star < C$, where $x^\star$ is the equilibrium rate of a controlled flow, and $y^\star$ is the

[3]Throughout this chapter, we use the term "uncontrolled flows" and "real-time flows" interchangeably.

mean rate of an uncontrolled flow. Further, we assume that each flow experiences a round trip delay d.

From the controlled flows' point of view, the system we have described above can be thought of as a closed loop system (with delay) and feedback control is applied at the router based on *aggregate arrivals*. A popular modeling and analysis methodology for such closed-loop systems in the Internet context has been through functional differential (or difference) equations based *fluid models* [22, 27].

The router is modeled by a *marking function* (see Section 4.3.2) which signals congestion by marking flows, and receivers detect the marks and inform the respective flow sources to increase or decrease their transmission rate. We model flows by discrete time fluid processes. We denote the fluid rates of individual flows by $\{x_k[i], k = 1, \ldots, n\}$, where $x_k[i]$ denotes the number of arrivals[4] of controlled flow k at time i. In this chapter, we consider two kinds of source rate adaptation algorithms: *(i) instant adaptation* and *(ii) weighted proportional fair controller*. Then, we represent the individual flow dynamics of each algorithm by [7, 24, 26, 64]:

Instant adaptation:

$$w \quad = \quad x_k[i]p\Big(\sum_{j=1}^{n} x_j[i] + \sum_{j=1}^{n} y_j[i], nC\Big) \tag{4.1}$$

In the instant adaptation algorithm, congestion controllers adapt to the fixed point of the difference equation in (4.2) with no delay [7, 64]. In this scheme, as the rate of the uncontrolled flow varies with time, the corresponding equilibrium rate varies appropriately (as determined by the elasticity properties of $p(\cdot)$), and the instant adaptation scheme tracks this variation of the equilibrium rate. This allows us to focus purely on the properties of the marking function and to ignore the effects of κ and d (described in the weighted proportional fair controller below).

[4]We use the terms "number of arrivals" and "arrivals" interchangeably. Further, the term "arrival rate" corresponds to the number of arrivals per time-slot.

Weighted proportional fair controller:

$$x_k[i+1] - x_k[i] \quad = \quad \kappa\Big(w - x_k[i-d]p\Big(\sum_{j=1}^{n} x_j[i-d] + \sum_{j=1}^{n} y_j[i-d], nC\Big)\Big), \tag{4.2}$$

where $y_k[i]$ denotes the number of arrivals of uncontrolled flow k at time i. κ and w are positive constants which determine the rate at which each flow increases or decreases its transmission rate, and the equilibrium point.

4.3.2 Marking Function

The marking function, $p(z, C)$ represents the fraction of flow to be marked when the total arrivals to the associated router with capacity C is z. We consider the following form of marking functions:

$$p(z, C) \quad = \quad \begin{cases} 0 & \text{if } 0 \le z \le \underline{m}, \\ \bar{p}(z, C) & \text{if } \underline{m} < z < \overline{m}, \\ 1 & \text{if } z \ge \overline{m}, \end{cases} \tag{4.3}$$

where $\underline{m} \in [0, C)$, $\overline{m} \in (0, \infty)$, and $\underline{m} < \overline{m}$. $\bar{p}(z, C)$ is assumed to satisfy the following condition.

Assumption 4.3.1. *We assume that $\bar{p}(z, C)$ is a increasing, Lipschitz continuous, differentiable function with range $[0, 1]$, that satisfies $\bar{p}(z, C) = \bar{p}(z/C, 1)$.*

Assumption 4.3.1 states that the fraction of packets marked depends only on the ratio of the total arrival rate and the link capacity, which is satisfied by typical marking functions such as those in Table 4.1 (see [9, 24] for more details)[5].

In Table 4.1, Type **M** has the interpretation of the queue length exceeding B in an M/M/1 queue with arrival rate z [24]. Type **R** can be used as a rate based model for REM (Random Exponential Marking [10]) for a suitable choice of α [4]. Type **L** is a linear marking function, and models a simplified form of RED (Random Early Detection [21]). Type **E** is

[5]For notational simplicity, we will omit the second parameter C throughout this chapter unless explicitly needed.

Table 4.1: Examples of Marking Functions

Type	M	R	L	E	V
$\hat{p}(z, C)$	$(\frac{z}{C})^B$	$\frac{\alpha z}{C-(1-\alpha)z}$	$\alpha(\frac{z}{C} - \eta)$	$1 - e^{-\frac{\alpha}{C}z}$	$\left(\frac{z-\alpha C}{z}\right)^+$
$< \underline{m}, \overline{m} >$	$< 0, C >$	$< 0, C >$	$< C\eta, C(1/\alpha + \eta) >$	$< 0, \infty >$	$< \alpha C, \infty >$

a rate based exponential marking. Finally, type **V** has the interpretation of the fraction of fluid lost when the arrival rate exceeds a certain level, called the "virtual capacity" [2].

Then, from Assumption 4.3.1, we present the individual flow dynamics at time i by

$$w = x_k[i]p\Big(\sum_{j=1}^{n} x_j[i] + \sum_{j=1}^{n} y_j[i], nC\Big) = x_k[i]p\Big(\frac{1}{n}\sum_{j=1}^{n} x_j[i] + \frac{1}{n}\sum_{j=1}^{n} y_j[i], C\Big). \tag{4.4}$$

By summing over the flow index k, we then have

$$w \quad = \quad x[i]p(x[i] + y[i]), \tag{4.5}$$

where $x[i]$ and $y[i]$ are *the average arrivals (over flows)* at time i, i.e.,

$$x[i] = \frac{1}{n}\sum_{j=1}^{n} x_j[i], \quad y[i] = \frac{1}{n}\sum_{j=1}^{n} y_j[i].$$

Similarly, with weighted proportional fair controller, we have

$$x[i+1] - x[i] \quad = \quad \kappa\Big(w - x[i-d]p(x[i-d] + y[i-d])\Big). \tag{4.6}$$

4.3.3 Elasticity of Marking Function: Warping

In this section, we describe how to parameterize the elasticity of marking functions by adopting "warped" marking functions. A warped marking function has a parameter (denoted by β), which determines the elasticity of the marking functions. *The family of warped marking functions enables us to alter the elasticity of the marking function without altering the steady-state utilization.* Prior to describing warping, we first make the following additional assumption on the marking function.

Assumption 4.3.2. *$1/p(z, C)$ is convex over $(\underline{m}, \infty)$.*

The typical marking functions in Table 4.1 satisfy Assumption 4.3.2.

Given any marking function $p(z)$ satisfying Assumption 4.3.1 and 4.3.2, we construct a family of marking functions $\{p_\beta(z),\ \beta \in (0, \infty)\}$, which are parameterized by β and are defined by

$$p_\beta(z) \quad \triangleq \quad p(f_\beta(z)),$$

where

$$f_\beta(z) \quad = \quad \begin{cases} \beta(z - \gamma) & \text{if} \quad 0 < \beta < 1, \\ \gamma z^\beta & \text{if} \quad \beta \geq 1. \end{cases}$$

For a given system (with a mixture of controlled and uncontrolled arrivals), let the equilibrium rate at the router be denoted by $z^\star$. For each value of β, the parameter γ (in the definition of $f_\beta(z)$) is chosen such that at this equilibrium rate $z^\star$, $f_\beta(z^\star) = z^\star$. This definition ensures that for each fixed nominal marking function, and the corresponding family of warped marking functions, $\{p_\beta(z)\}$, $\beta \in (0, \infty)$, *the steady-state utilization of the system is independent of* β. Then, for $z > z^\star$, we have $p_\beta(z) > p(z)$, if $\beta > 1$, and $p_\beta(z) < p(z)$ if $0 < \beta < 1$.

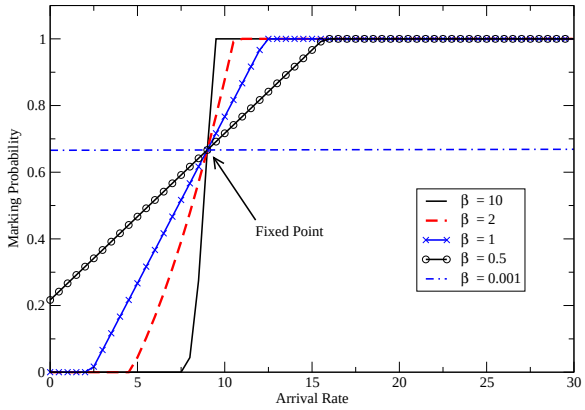
In other words, $\{p_\beta(z)\}$ corresponds to a family of marking functions whose elasticity is varying with respect to the nominal marking function $p(z)$[6] (see Figure 4.4 for examples). If $\beta > 1$, $p_\beta(z)$ is *more elastic*, and if $\beta < 1$, $p_\beta(z)$ is *less elastic* from Definition 4.2.1. We can easily check that for each $\beta > 0$, $p_\beta(z)$ satisfies Assumption 4.3.1 and 4.3.2.

Table 4.2 provides the warped marking functions (expressed in terms of $x^\star$ and $z^\star$) for the example marking functions in Table 4.1 for different values of β. As an example, consider the Type **E** marking function, $p(z) = 1 - e^{-\alpha z/C}$, and suppose that $\beta > 1$. Then, it is not difficult to observe that the warped marking function is $1 - (1 - \frac{w}{x^\star})^{(\beta \frac{z}{z^\star} + 1 - \beta)}$ using the following two conditions: *(i)* $p(z^\star) = p(f_\beta(z^\star))$, and *(ii)* $w = x^\star p(z^\star)$, which follows from the equilibrium analysis in (4.5) and (4.6). Figure 4.4 shows the warped marking functions of

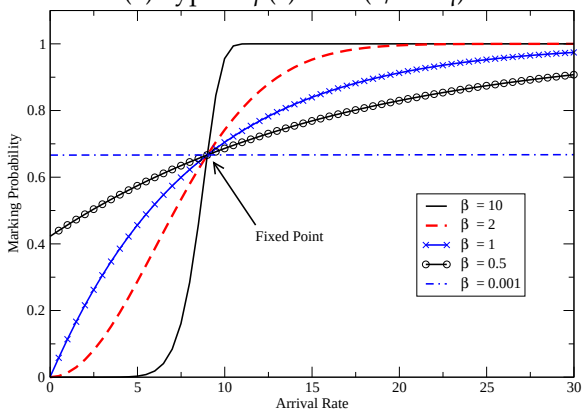
[6]When $\beta = 1$, we use $p_\beta(\cdot)$ and $p(\cdot)$ interchangeably throughout this paper.

Table 4.2: Examples of Warped Marking Function expressed in terms of w, $x^\star$, and $z^\star$

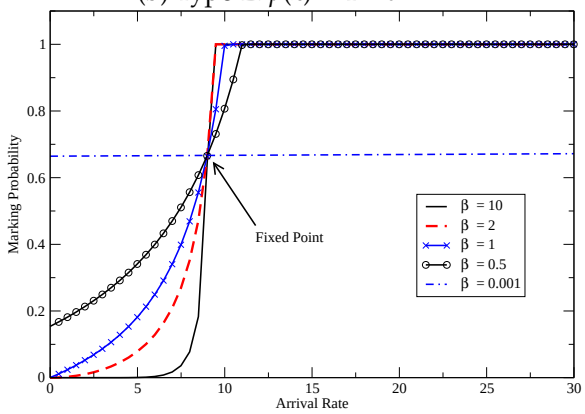
Marking Func	$\beta = 1$	$0 < \beta < 1$	$\beta > 1$
$\alpha(\frac{z}{C} - \eta)$	$\frac{w}{x^\star} + \alpha z^\star(\frac{z}{z^\star} - 1)/C$	$\frac{w}{x^\star} + \beta\alpha z^\star(\frac{z}{z^\star} - 1)/C$	$\frac{w}{x^\star} + \alpha z^\star((\frac{z}{z^\star})^\beta - 1)/C$
$(\frac{z}{C})^B$	$(\frac{w}{x^\star})(\frac{z}{z^\star})^{(\log \frac{w}{x^\star})/(\log \frac{z^\star}{C})}$	$(\frac{w}{x^\star})(\beta\frac{z}{z^\star} + 1 - \beta)^{(\log \frac{w}{x^\star})/(\log \frac{z^\star}{C})}$	$(\frac{w}{x^\star})(\frac{z}{z^\star})^{\beta(\log \frac{w}{x^\star})/(\log \frac{z^\star}{C})}$
$\dfrac{\alpha z}{C - (1-\alpha)z}$	$\dfrac{w(C - z^\star)(\frac{z}{z^\star})}{C(x^\star - w) - (z^\star x^\star - wC)(\frac{z}{z^\star})}$	$\dfrac{w(C - z^\star)(\beta\frac{z}{z^\star} + 1 - \beta)}{C(x^\star - w) - (z^\star x^\star - wC)(\beta\frac{z}{z^\star} + 1 - \beta)}$	$\dfrac{w(C - z^\star)(\frac{z}{z^\star})^\beta}{C(x^\star - w) - (z^\star x^\star - wC)(\frac{z}{z^\star})^\beta}$
$1 - e^{-\alpha\frac{z}{C}}$	$1 - (1 - \frac{w}{x^\star})^{(\frac{z}{z^\star})}$	$1 - (1 - \frac{w}{x^\star})^{(\beta\frac{z}{z^\star} + 1 - \beta)}$	$1 - (1 - \frac{w}{x^\star})^{(\frac{z}{z^\star})^\beta}$
$\left(\dfrac{z - \alpha C}{z}\right)^+$	$\left(1 - \dfrac{1 - \frac{w}{x^\star}}{\frac{z}{z^\star}}\right)^+$	$\left(1 - \dfrac{1 - \frac{w}{x^\star}}{\beta(\frac{z}{z^\star}) + 1 - \beta}\right)^+$	$\left(1 - \dfrac{1 - \frac{w}{x^\star}}{(\frac{z}{z^\star})^\beta}\right)^+$



(a) Type **L**: $p(z) = 0.1(z/C - \eta)$



(b) Type **E**: $p(z) = 1 - e^{-\alpha z/C}$



(c) Type **R**: $p(z) = \frac{\alpha z}{C-(1-\alpha)z}$

Figure 4.4: Examples of Warped Marking Functions: $C = 10$, $w = 3$, and $z^\star = 9$.

Type **L**, Type **E**, and Type **R** for different values of β.

4.3.4 Problem Statement

Our objective is to study the effect of the marking elasticity (using the warped marking functions) on the QoS of uncontrolled flows. A widely used QoS parameter (for the uncontrolled real-time flows) is the probability that the queue length exceeds a fixed threshold. It is clear that the QoS performance for uncontrolled real-time flows will be the "best" if such flows are always given strict priority access at the routers (i.e., priority scheduling at the router with priority for uncontrolled real-time flows). We will later use priority scheduling as a reference model to assess the performance of FIFO scheduling (used in Section 4.3.3 to study scheduling-elasticity trade-off). With priority scheduling, we assume that two separate queues are used to store data from the controlled and uncontrolled flows, respectively.

We denote the sum of arrivals of n uncontrolled and n controlled flows over the time interval $[i, j)$ by $Y^n[i, j) = \sum_{k=1}^{n} \sum_{s=i}^{j-1} Y_k[s]$ and $X^n[i, j) = \sum_{k=1}^{n} \sum_{s=i}^{j-1} X_k[s]$, respectively[7]. We let $Z^n[i, j) = Y^n[i, j) + X^n[i, j)$, to denote the total sum of controlled and uncontrolled arrivals over the same time interval $[i, j)$ in the n-th system.

We consider a discrete time framework, where we suppose that the current time is 0, and the arrival process starts at time $-\infty$. Thus, at the current time, the system is in steady state. We denote the queue length at time 0 with FIFO and priority schedulers by Q_0^P (for a queue of uncontrolled real-time flows) and Q_0^F, respectively.

The steady-state behavior of the Internet congestion controllers (i.e., routers accessed by a mixture of controlled and uncontrolled flows) has been studied under fluid models [9, 12], and stability condition has been established [7, 9]. However, our focus here in the transient behavior which leads to queue overflow, and the impacts on the QoS of real-time flows. Thus, in this chapter, our objective is to compute the queue overflow probability as a function of the time-scale of the transient phenomenon as well as the marking function elasticity. We assume that for a fixed finite T_I, the system is stable before $-T_I$, i.e., $x[i] = x^\star$ and $y[i] = y^\star$ for $\forall i < -T_I$; and thus, the queue over flow probability is a function of T_I, the marking function, and scheduling policy. By the queue stability assumption (i.e.,

[7]Thus, $X_k[s]$ denotes the random variable corresponding to the number of arrivals from the k^{th} controlled flow at time s, and a similar definition holds for $Y_k[s]$. Finally, we use upper-case letters and lower-case letters to denote random variables and deterministic quantities, respectively.

$x^\star + y^\star < C$), the queue length at time $i \in [-\infty, -T_I)$ is 0. Therefore, it suffices to consider the arrival processes only at the time interval over $[-T_I, 0)$.

For a fixed T_I, consider a following non-negative *scaled* (deterministic) arrival vector over the interval $[-T_I, 0)$: $\vec{v}[-T_I, 0) = \big(v[-T_I], v[-T_I + 1], \ldots, v[-1]\big)$. Then, from Loyne's formula on the queue length process, the queue length at time 0 corresponding to an arrival vector $\vec{v}[-T_I, 0)$ can be defined as:

$$Q\big(\vec{v}[-T_I, 0)\big) \triangleq \left[\sup_{0<T\le T_I} \Big(\sum_{i=-T}^{-1} v[i] - CT\Big)\right]^+ \tag{4.7}$$

Thus, the queue overflows probabilities of priority and FIFO queueing are given by:

$$\begin{aligned}
\Pr(Q_0^P \ge nB) &= \Pr\left[Q\Big(\frac{1}{n}Y^n[-T_I, 0)\Big) \ge B\right] \\
&= \Pr\left[\sup_{0<T\le T_I} \Big(\frac{1}{n}Y^n[-T, 0) - CT\Big) \ge B\right] \qquad (4.8)\\
\Pr(Q_0^F \ge nB) &= \Pr\left[Q\Big(\frac{1}{n}Z^n[-\infty, 0)\Big) \ge B\right] \\
&= \Pr\left[\sup_{0<T\le T_I} \Big(\frac{1}{n}Z^n[-T, 0) - CT\Big) \ge B\right]. \qquad (4.9)
\end{aligned}$$

In the large n regime, we derive asymptotic expressions for the queue overflow probabilities using large deviation techniques, which requires the application of the contraction principle [70]. Applicability of contraction principle depends on the continuity of the queue length at time 0 with respect to *the arrival process from the uncontrolled flows*.

With the instant adaptation, we define the queue length at time 0 for the uncontrolled arrival $\vec{y}[-T_I, 0)$ by:

$$\widetilde{Q}\big(\vec{y}[-T_I, 0)\big) \triangleq \left[\sup_{0<T\le T_I} \Big(\sum_{i=-T}^{-1} (x[i] + y[i]) - CT\Big)\right]^+, \tag{4.10}$$

where $\vec{x}[-T_I, 0)$ is determined by (4.5). Similarly, we use the notation $\hat{Q}$ to refer to the queue length at time 0 with the weighted proportional fair controller, i.e, $\vec{x}[-T_I, 0)$ is determined by (4.6).

In Section 4.4, with the instant adaptation, we first prove that $\widetilde{Q} : \mathcal{R}^{T_I} \mapsto \mathcal{R}$ is continuous

under a suitable topology (which allows the application of a LDP (large deviation principle) and the contraction principle) in Theorem 4.4.1. Next, by computing Q_0^F as a function of marking elasticity, we will investigate and quantify stability-elasticity and scheduling-elasticity trade-off. In section 4.5, we extend the analysis to the weighted proportional fair controller (using the queue length function $\hat{Q} : \mathcal{R}^{T_I} \mapsto \mathcal{R}$).

4.4 Instant Adaptation Controller

In this section, with instant adaptation controller, we study the effect of elasticity of the marking function on the QoS guarantees for uncontrolled flows by deriving (upper and lower) bounds on the queue overflow probability. Through this study, we discuss stability-elasticity and scheduling-elasticity trade-off.

4.4.1 Continuity of Queue Length and Queue Overflow Probability

We first prove the continuity of $\widetilde{Q}$ (with respect to the uncontrolled arrival process) that enables the application of the large deviation results and the contraction principle.

We present a useful lemma, which is used to derive the queue overflow probability and to prove that the queue length at time 0 is continuous with respect to the uncontrolled arrival process. With w and $p(\cdot)$ in (4.5), let us define a function $h : [\underline{z}, \infty) \mapsto \mathcal{R}^+ \cup \{0\}$, where

$$h(z) \triangleq z - \frac{w}{p(z)}, \quad \underline{z}p(\underline{z}) = w.$$

Since $p(\underline{z}) > 0$, we have $\underline{z} > \underline{m}$ from (4.3) on $p(z)$. Thus, $h(z)$ is defined over $[\underline{z}, \infty)$. Also, from the definition of $h(z)$, $h(\underline{z}) = 0$ and $h(z) > 0$ when $z > \underline{z}$. Intuitively, z is the total average arrival rate over flows (i.e., the sum of average uncontrolled and controlled arrivals over flows) at time i, and $h(z)$ is the corresponding uncontrolled arrival rate at time i (see (4.5)).

Lemma 4.4.1. *Suppose that we have a marking function of the form (4.3) satisfying Assumption 4.3.1 and 4.3.2. Then, h is invertible and concave. Further, by defining $g \triangleq h^{-1}$, g is a convex function satisfying*

$$\Big(g(y[i]) - y[i]\Big)p\Big(g(y[i])\Big) = w$$

Proof. Observe that $g(u)$ is the average total sum of uncontrolled and controlled arrivals with respect to the uncontrolled arrival u. Note that since $h(\cdot)$ is continuous, $g(\cdot)$ is also continuous. The formal proof is presented in the Appendix. □

Definition 4.4.1 (Uniform Norm). *We define the* uniform norm *[71] for a vector $\vec{v}[-T_I, 0)$ as follows:*

$$\|\vec{v}[-T_I, 0)\|_u \triangleq \sup_{0<T\leq T_I} \left| \frac{\sum_{i=-T}^{i=-1} v[i]}{T} \right|$$

We now prove the following result:

Theorem 4.4.1. *The queue length function (at time 0) $\widetilde{Q} : \mathcal{R}^{T_I} \mapsto \mathcal{R}$ is continuous with respect to the uncontrolled arrival process $\vec{y}[-T_I, 0) \in \mathcal{R}^{T_I}$ in the topology endowed with uniform norm. Further, we have*

$$\lim_{n\to\infty} \frac{1}{n} \log Pr\left(Q\Big(\frac{1}{n} Z^n[-T_I, 0)\Big) \geq B\right) = -I_F(B), \tag{4.11}$$

where

$$\begin{aligned} I_F(B) &\triangleq \inf_{0<T\leq T_I} \inf_{\vec{y}[-T,0):\widetilde{Q}(\vec{y}[-T,0))\geq B} \mathbf{I}\Big(\vec{y}[-T, 0)\Big) \\ \mathbf{I}\Big(\vec{y}[-T, 0)\Big) &\triangleq \sup_{\vec{\theta}\in\mathcal{R}^T} \left(\vec{\theta}\cdot\vec{y}[-T, 0) - \log E e^{\vec{\theta}\cdot\vec{Y}[-T,0)}\right), \end{aligned} \tag{4.12}$$

where $\cdot$ is the inner product of two vectors.

Proof. The proof is presented in the Appendix. □

4.4.2 Computation of Bounds on the Rate Function

This section focuses on computation of lower and upper bound on $I_F(B)$, leading to upper and lower bound on asymptotic queue overflow probability, respectively (see (4.11)). First, we add an additional assumption that an uncontrolled flows are independent and identically distributed *over time* for computational simplicity. The computation of $I_F(B)$ for non-i.i.d arrivals is left as future work. This i.i.d assumption ensures [70] that for any fixed

T, we have

$$\mathbf{I}(\vec{y}[-T,0)) \quad = \quad \sum_{i=-T}^{-1} I(y[i]), \tag{4.13}$$

where $I(y) \triangleq \sup_\theta \left(y\theta - \log \mathrm{E}(e^{\theta Y_1[-1]})\right)$, and $Y_1[-1]$ is the random variable denoting the number of arrivals from flow '1' at time slot '-1'.

From Theorem 4.4.1 and (4.13), the rate function is given by:

$$I_F(B) \quad = \quad \inf_{0<T\le T_I} I_F^T(B), \tag{4.14}$$

where

$$I_F^T(B) = \inf_{\vec{y}[-T,0)\in\mathcal{A}} \sum_{i=-T}^{-1} I(y[i]), \quad \mathcal{A} = \left\{\vec{a}[-T,0) : \widetilde{Q}(\vec{a}[-T,0)) \ge B\right\}.$$

Then, we have the following result on the upper and lower bound on $I_F(B)$.

Theorem 4.4.2 (Upper and lower bound).

$$\inf_{0<T\le T_I} TI\Big(C + \frac{B}{T} - \frac{w}{T}\Big(\frac{1}{p(B+C)} + \frac{(T-1)}{p(C)}\Big)\Big) \le I_F(B) \le \inf_{0<T\le T_I} TI\left(C + \frac{B}{T} - \frac{w}{p(C+B/T)}\right)$$

We first describe three useful lemmas that will be used in the proof of Theorem 4.4.2. All the proofs of these lemmas are presented in the Appendix.

First, we show that we do not need to optimize over the entire trajectories in the space $\mathcal{A}$. Instead, it is enough to optimize over the trajectories over $\hat{\mathcal{A}} \subset \mathcal{A}$, defined by

$$\hat{\mathcal{A}} \quad = \quad \left\{\vec{a}[-T,0) : \sum_{i=-T}^{-1} g(a[i]) \ge B + CT, a[i] \ge \hat{a},\ \forall i \in [-T,-1],\ g(\hat{a}) = C\right\}. \tag{4.15}$$

Recall that $g(u)$ is the average total sum of uncontrolled and controlled arrivals with respect to the uncontrolled arrival u.

Lemma 4.4.2. *With $\hat{\mathcal{A}}$ defined as (4.15),*

$$\inf_{0<T\leq T_I}\inf_{\vec{y}[-T,0)\in\mathcal{A}}\sum_{i=-T}^{-1}I(y[i]) \quad = \quad \inf_{0<T\leq T_I}\inf_{\vec{y}[-T,0)\in\hat{\mathcal{A}}}\sum_{i=-T}^{-1}I(y[i]) \tag{4.16}$$

Proof. The proof is presented in the Appendix. □

Next, let us define the space $\mathcal{B}$ given by:

$$\mathcal{B}=\Big\{\vec{a}[-T,0):\frac{1}{T}\sum_{i=-T}^{-1}a[i]\geq g^{-1}\Big(\frac{B}{T}+C\Big),\ a[i]\geq\hat{a},\ \forall i\in[-T,-1],\ g(\hat{a})=C\Big\}$$

By showing that the space $\mathcal{B}\subset\hat{\mathcal{A}}$, we derive an upper bound of $I_F^T(B)$, described in the following lemma.

Lemma 4.4.3. *$\mathcal{B}\subset\hat{\mathcal{A}}$. Thus,*

$$\inf_{\vec{y}[-T,0)\in\hat{\mathcal{A}}}\sum_{i=-T}^{-1}I(y[i]) \quad \leq \quad \inf_{\vec{y}[-T,0)\in\mathcal{B}}\sum_{i=-T}^{-1}I(y[i])$$

Proof. The proof is presented in the Appendix. □

Lemma 4.4.2 and Lemma 4.4.2 are used to derive the upper bound in Theorem 4.4.2. Finally, we describe a lemma, which will be used to derive the lower bound in Theorem 4.4.2.

Lemma 4.4.4. *Suppose that we have the following optimization problem.*

$$\inf_{\vec{z}[-T,0)\in C}\frac{1}{T}\sum_{i=-T}^{-1}f(z[i]),\quad C=\Big\{\vec{a}[-T,0):\sum_{i=-T}^{-1}a[i]\geq B+CT,\ a[i]\geq C\Big\}, \tag{4.17}$$

and f is increasing and concave. Then, the vector $\vec{z}^{\star}[-T,0)=(B+C,C,\ldots,C)$ is an optimizer.

Proof. The proof is presented in the Appendix. □

Proof. (i) Upper bound: From Lemma 4.4.2 and Lemma 4.4.3,

$$I_F(B) \quad = \quad \inf_{0<T\leq T_I}\inf_{\vec{y}[-T,0)\in\hat{\mathcal{A}}}\sum_{i=-T}^{-1}I(y[i]) \ \leq \ \inf_{0<T\leq T_I}\inf_{\vec{y}[-T,0)\in\mathcal{B}}I\Big(\frac{1}{T}\sum_{i=-T}^{-1}a[i]\Big)$$

Since $I(x)$ is increasing and convex, for $x \geq \hat{a}$, and from the definition of $\mathcal{B}$ (which is a convex set), we have the optimum when $a[i] = g^{-1}(B/T + C)$ for $-T \leq i < 0$. Then, the result immediately follows.

(ii) Lower bound: Since $I(x)$ is monotone-increasing over $[\hat{a}, \infty)$ and from Jensen's inequality, we have

$$\begin{aligned} I_F(B) &= \inf_{0<T\leq T_I} \inf_{\vec{y}[-T,0)\in\mathcal{A}} \sum_{i=-T}^{-1} I(y[i]) \\ &\geq \inf_{o<T\leq T_I} \inf_{\vec{y}[-T,0)\in\mathcal{A}} TI\Big(\frac{1}{T}\sum_{i=-T}^{-1} y[i]\Big) \\ &\geq \inf_{0<T\leq T_I} TI\Big(\inf_{\vec{y}[-T,0)\in\mathcal{A}} \frac{1}{T}\sum_{i=-T}^{-1} y[i]\Big) \end{aligned}$$

Let $z[i] = g(y[i])$ (i.e., $h(z[i]) = y[i]$ from Lemma 4.4.1). Then, we have the following problem transformation:

$$\inf_{\vec{y}[-T,0)\in\mathcal{A}} \frac{1}{T}\sum_{i=-T}^{-1} y[i] \quad \Leftrightarrow \quad \inf_{\vec{z}[-T,0)\in C} \frac{1}{T}\sum_{i=-T}^{-1} h(z[i]),$$

where C is defined as (4.17). Then, from Lemma 4.4.4 and the definition of h, the result follows. □

4.4.3 Stability-Elasticity Trade-off

Using the lower and upper bounds on the rate function (i.e., $I_F(B)$) derived in the previous section with the instant adaptation source controller, we study the effect of elasticity of marking functions on the stability (for controlled flows) and QoS (for uncontrolled flows), and their trade-off.

Consider a fixed nominal marking function $p(z)$, and the corresponding family of marking functions $\{p_\beta(z),\ \beta \in (0, \infty)\}$ Recall that with respect to a nominal marking function, $\beta > 1$ corresponds to a more elastic marking function, and $\beta < 1$ corresponds to a less elastic marking function.

We adopt the following procedure to study the effect of the marking elasticity on the system stability of the closed-loop controlled sources and on the QoS of the open-loop

uncontrolled real-time sources:

(i) For a fixed β, we compute the *best* QoS that can be achieved for the uncontrolled real-time sources by assuming that the controlled source adapts and backs-off instantly in response to congestion, i.e., the instant adaptation controller is used for QoS analysis. Note that with the instant adaptation scheme, the upper bound on the rate function from Theorem 4.4.2 provides a lower bound on the queue overflow probability. In other words, for a fixed value of β and the corresponding marking function $p_\beta(z)$, we can get *no better QoS* than that given by Theorem 4.4.2.

(ii) Using known local stability results for a weighed proportional fair controller from [7], for a fixed β, we compute the *maximum delay* that can be tolerated before the controlled sources go into local (and hence, global) instability.

While *(i)* and *(ii)* use different controllers (instant adaptation and proportional fair controller, respectively), our objective here is to illustrate the effect of the marking elasticity (not specific controller mechanisms) on the QoS for the uncontrolled real-time source. Thus, *(i)* corresponds to the "best-case" scenario for the QoS of the uncontrolled real-time sources (due to the fact that the controlled flows in *(i)* back-off instantly). With any other controller, there will be a lag associated with the back-off of controlled flows, thus resulting in poorer QoS for uncontrolled real-time flows than that with the instant adaptation controller [8].

To discuss the stability analysis in *(ii)*, we use the local stability condition for a weighted proportional fair controller discussed in (4.2). For each marking function $p_\beta(z)$, we determine the *maximum round-trip propagation delay d* that the system can tolerate before going into local instability (and thus, global instability). This is given by [7]:

$$\kappa(p_\beta(z^\star) + z^\star {p'}_\beta(z^\star)) \quad < \quad \sin\left(\frac{\pi}{2(2d+1)}\right) \tag{4.18}$$

Further, by definition of $p_\beta(x)$, we have

$${p'}_\beta(z^\star) \;=\; {f'}_\beta(z^\star)p'(z^\star) \;=\; \beta p'(z^\star).$$

[8] See Section 4.5 for the corresponding result when the weighted proportional fair is used for both *(i)* (QoS analysis) and *(ii)* (stability analysis)

Thus, for each value of β, the stability condition (4.18) reduces to

$$\kappa(p(z^\star) + \beta z^\star p'(z^\star)) \quad < \quad \sin\left(\frac{\pi}{2(2d+1)}\right) \tag{4.19}$$

The trade-off between the QoS for the real-time sources and the stability for the controlled sources is parameterized by β, the elasticity of the marking function. The more elastic the marking function is, the worse is the stability behavior (as β becomes larger in (4.19)). On the other hand, increasing β *improves* the QoS behavior for the real-time uncontrolled flows. This can be explained by applying the β-elastic marking function $p_\beta(\cdot)$ to the lower and upper bound of the rate function in Theorem 4.4.2, and by observing that $p_\beta(B+C)$, $p_\beta(C)$, and $p_\beta(C+B/T)$ are increasing with respect to β. For this reason, we refer to this study as *stability-elasticity trade-off*. See Figure 4.5 and Figure 4.9 for numerical examples of the stability-elasticity trade-off with both the instant adaptation and the weighted proportional fair controller, respectively, under various marking functions and network environments.

4.4.4 Scheduling-Elasticity Trade-off

In this section, we derive bounds on the link capacities needed with priority and FIFO scheduling to support a QoS requirement for the real-time uncontrolled flows, which stipulates that the queue overflow probability should not exceed some ϵ.

It is clear that the link capacity required for supporting a fixed queue overflow probability with priority scheduling is the smallest (over all scheduling policies), since absolute priority is given to these real-time flows, i.e., the controlled flows do not affect the queue dynamics for the uncontrolled flows (see Figure 4.1). Thus, the required capacity with priority scheduling does not depends on the marking elasticity. On the other hand, with FIFO scheduling, the behavior of uncontrolled flows and controlled flows are coupled together, and thus, the required link capacity for supporting the given QoS is *a function of marking elasticity*, and it will be larger than that with priority scheduling.

With this observation, our objective is to quantitatively study the "scheduling-gain" (see Definition 4.4.2) of priority scheduling over FIFO scheduling, as a function of marking

elasticity, and we show that this gain could be significantly reduced by increasing marking elasticity.

To do so, we adopt the following approach:

(i) For a fixed marking elasticity, we first determine the per-flow link capacities needed with priority and FIFO scheduling (denoted by C_P and $C_F(\beta)$, respectively) for supporting a fixed queue overflow probability ϵ.

(ii) Using the analysis in *(i)* we define the following "normalized scheduling-gain" of priority scheduling over FIFO scheduling:

Definition 4.4.2 (Scheduling Gain).

$$\Delta_C(\beta) \triangleq \frac{C_F(\beta) - C_P}{C_P} \tag{4.20}$$

Intuitively, $\Delta_C(\beta)$ quantifies the trade-off between the penalty of choosing "sub-optimal" scheduling algorithm (i.e., FIFO scheduling) in terms of QoS-guarantee and the elasticity of the marking function (i.e., β). We will investigate the behavior of $\Delta_C(\beta)$ as the marking elasticity parameter, β, changes.

To discuss the analysis with priority scheduling in *(i)* (i.e., computation of the bound on C_P), we use well-known large deviation results [66]. In the large number of flows regime, defining $\delta \triangleq -\frac{1}{n}\log\epsilon$, the system would allow the queue overflow probability less than ϵ if $I_P(B) \leq \delta$, where

$$I_P(B) = \inf_{0<T\leq T_l} TI\Big(C_P + \frac{B}{T}\Big).$$

Observe that this also leads to an *effective bandwidth* characterization for a single server queue accessed by only uncontrolled flows [65,66], where a sufficient condition for $I_P(B) \geq \delta$ (equivalently, to support a queue overflow probability of at most $e^{-n\delta}$) is

$$\frac{\Lambda(\delta/B)}{\delta/B} \leq C_P, \tag{4.21}$$

where $\Lambda(\theta)$ is the log-moment generating function of a random uncontrolled arrival at a

particular time-slot, i.e., $\Lambda(\theta) = \log \mathrm{E}[e^{\theta Y_1[-1]}]$.

Next, for the analysis with FIFO scheduling (i.e., computation of $C_F(\beta)$) in *(i)*, we have the following proposition:

Proposition 4.4.1. *With FIFO scheduling and with a fixed marking elasticity β, a sufficient condition for $I_F(B) \geq \delta$ (i.e., to support the queue overflow probability ϵ) is:*

$$\frac{\Lambda(\delta/B)}{\delta/B} \leq C_F(\beta) - \frac{w}{p_\beta(C_F(\beta))}. \tag{4.22}$$

Proof. The proof is presented in the Appendix. □

We now apply (4.21) and Proposition 4.4.1 to study the effect of marking elasticity on the scheduling gain of priority scheduling (discussed in *(ii)*). Defining $g(\beta) \triangleq C_F(\beta) - \frac{w}{p_\beta(C_F(\beta))}$, we observe that $g(\beta)$ is increasing in β, since $p_\beta(z)$ is increasing in β for $z \geq z^\star$, and $C_F(\beta) \geq z^\star$. This implies that we need progressively smaller $C_F(\beta)$ (to support a given QoS for the uncontrolled real-time flows and queue stability) with increasing values of β.

Further, from the definition of warped marking function discussed in Section 4.3.3, for a fixed $z > z^\star$, we have $p_\beta(z) \to p(z^\star)$, as $\beta \to 0$, and $p_\beta(z) \to 1$, as $\beta \to \infty$. Thus, from (4.22), we have

$$C_F(\beta) \overset{\beta \to 0}{\longrightarrow} \frac{\Lambda(\delta/B)}{\delta/B} + \frac{w}{p(z^\star)}, \qquad C_F(\beta) \overset{\beta \to \infty}{\longrightarrow} \frac{\Lambda(\delta/B)}{\delta/B} + w \tag{4.23}$$

Note that $\frac{\Lambda(\delta/B)}{\delta/B}$ is the minimum required per-flow link capacity with priority scheduling for supporting the given queue overflow probability ϵ (which follows from (4.21)). Then, the results in (4.23) imply that by increasing the marking elasticity, the scheduling gain can be significantly reduced. We illustrate this by means of an example.

Consider a single bottleneck network accessed by 100 uncontrolled and controlled flows (i.e., $n = 100$). Also, let the equilibrium rate for a controlled flow and the mean rate of an uncontrolled flow to be 640 kbps and 120 kbps, respectively. Assuming that each packet is of a fixed size with 1000 bytes, the equilibrium rate of a controlled flow is $x^\star = 80$ pkts/sec, and the mean rate of uncontrolled flow is $y^\star = 15$ pkts/sec. We model each uncontrolled arrival process by a ON-OFF process with 'ON' probability of 0.1, and 'ON' rate of 150

pkts/sec. The queue buffer size is 800 kbytes, which corresponds to 100 pkts (i.e., $B = 1$). We set the equilibrium marking probability to be 0.03 (i.e., $p(z^\star) = p(95) = 0.03$). The QoS parameter (queue overflow probability) for uncontrolled flows is set to be $\epsilon = 10^{-6}$ (i.e., $\delta = -\frac{1}{100}\log 10^{-6} = 0.1382$).

Then, the required per-flow link capacity with priority scheduling is $\frac{\Lambda(0.1382/1)}{0.1382/1} = 133.3$ pkts/sec. From the equilibrium analysis of the congestion controller, we have $w = x^\star \times p(z^\star) = 80 \times 0.03 = 2.4$. Then, with FIFO scheduling, to support the same queue overflow probability ϵ, we need the per-flow link capacity $133.3 + \frac{2.4}{0.03} = 213.3$ pkts/sec for $\beta = 0$. On the other hand, for $\beta \to \infty$, we need only 133.3 + 2.4 = 135.7 pkts/sec.

In terms of queue management and implementation, FIFO scheduling is much simpler than priority scheduling. On the other hand, it is clear that priority scheduling provides better QoS to the uncontrolled real-time flows. However, the results in this section imply that the scheduling gain due to priority scheduling may not become significant in the large scale networks by adjusting the marking elasticity. In other words, the difference in the required capacities with FIFO and priority scheduling for a fixed QoS (queue overflow probability) can be significantly reduced by increasing the marking elasticity. For this reason, we refer to this as *scheduling-elasticity trade-off*. This trade-off is graphically illustrated in Figure 4.7 and Figure 4.8 in Section 4.6, where we numerically present the trade-off.

4.5 Weighted Proportional Fair Controller

Thus far, we have considered the effect of elasticity of marking functions on the QoS guarantee and the stability using instant adaptation. In this section, we consider the weighted proportional fair controller described in (4.2). As in Section 4.4, we again assume that the uncontrolled arrivals are independent over time. However, the analysis with the weighted proportional fair controller is more complicated than that with the instant adaptation due to temporal coupling of the arrival process introduced by the dynamics of each congestion controller, i.e., the total arrivals to the router are not independent over time, *even if* the uncontrolled arrivals are. We begin this section with the proof of continuity of queue length (at time 0) function (i.e., $\hat{Q}(\cdot)$) with respect to the uncontrolled arrival process

in the system with controlled flows governed by weighted proportional fair controller, and using this, derive an upper-bound on the rate function to study the stability-elasticity trade-off.

4.5.1 Continuity of Queue Length and Queue Overflow Probability

We have the following theorem to show that the queue length at time 0 is continuous with weighted proportional fair controller and round-trip propagation delay d.

Theorem 4.5.1. *The queue length function (at time 0) $\hat{Q} : \mathcal{R}^{T_I} \mapsto \mathcal{R}$ is continuous with respect to the uncontrolled arrival process $\vec{y}[-T_I, 0)$ in the topology endowed with uniform norm. Thus, we have*

$$\lim_{n\to\infty} \frac{1}{n} \log Pr\Big(Q\big(\frac{1}{n}Z^n[-T_I, 0)\big) \geq B\Big) \quad = \quad -I_{\hat{F}}(B),$$

where

$$I_{\hat{F}}(B) \quad \triangleq \quad \inf_{0<T\leq T_I} \inf_{\vec{y}[-T,0):\hat{Q}(\vec{y}[-T,0))\geq B} \mathbf{I}\big(\vec{y}[-T, 0)\big) \tag{4.24}$$

where $\mathbf{I}\big(\vec{y}[-T, 0)\big)$ has been defined in Theorem 4.4.1.

Proof. The proof is presented in the Appendix. □

4.5.2 Stability-Elasticity Trade-off

In the previous section, we have shown that the continuity of queue length with respect to the uncontrolled arrival process. This enables us to apply the contraction principle as in Section 4.4.2. In this section, we briefly study the stability-elasticity trade-off by deriving an upper bound on the rate function in the system with a "one-step" delay. Controllers with delay and the scheduling-elasticity trade-off will be studied in Section 4.6 using simulations.

In our analysis, we assume that the controller gain, κ, is small enough to prevent the transmission rate of a controlled flow from becoming negative (note that with a one-step delay, if κ is large, the controller will be unstable irrespective of the marking elasticity).

Theorem 4.5.2 (Upper bound). *Suppose that $\kappa p(C+B) \leq 1$. Then, we have*

$$I_{\hat{F}}(B) \quad \leq \quad \inf_{0<T\leq T_I} \sum_{i=-T}^{-1} I\left(C + \frac{B}{T} - \Big(x^\star - \frac{w}{\hat{p}(T)}\Big)(1-k\hat{p}(T))^{i+T+1} - \frac{w}{\hat{p}(T)}\right),$$

where $\hat{p}(T) = p(C + B/T)$.

Proof. First, from Theorem 4.5.1, we have

$$I_{\hat{F}}(B) \quad = \quad \inf_{0<T\leq T_I} \inf_{\vec{y}[-T,0):\hat{Q}(\vec{y}[-T,0))\geq B} \sum_{i=-T}^{-1} I(y[i]) \tag{4.25}$$

Next, since for a fixed T, $\{\vec{y}[-T,0) \mid x[i] + y[i] = C + B/T,\ i \in [-T,-0)\} \subset \{\vec{y}[-T,0) \mid \hat{Q}(\vec{y}[-T,0)) \geq B\}$, we have

$$I_{\hat{F}}(B) \quad \leq \quad \inf_{0<T\leq T_I} \inf_{\vec{y}[-T,0):y[i]+x[i]=C+B/T, i\in[-T,0)} \sum_{i=-T}^{-1} I(y[i]) \tag{4.26}$$

Then, from (4.6), the fact that $y[i] + x[i] = C + B/T,\ \forall i \in [-T,0)$ implies that $\vec{x}[-T,0)$ is deterministically fixed by the following form of difference equation:

$$x[i+1] - x[i] \quad = \quad \kappa\Big(w - x[i]\hat{p}(T)\Big), \quad i \in [-T,0), \tag{4.27}$$

where the initial condition $x[-T-1] = x^\star$. Then, by solving the difference equation (4.27), and by using $x[i] + y[i] = C + B/T, \forall i \in [-T,0)$ we have

$$\begin{aligned} x[i] \quad &= \quad \Big(x^\star - \frac{w}{\hat{p}(T)}\Big)\Big(1-\kappa\hat{p}(T)\Big)^{i+T+1} + \frac{w}{\hat{p}(T)} \\ y[i] \quad &= \quad C + \frac{B}{T} - \Big(x^\star - \frac{w}{\hat{p}(T)}\Big)(1-\kappa\hat{p}(T))^{i+T+1} - \frac{w}{\hat{p}(T)}. \end{aligned}$$

The assumption that $\kappa p(C+B) < 1$ ensures that $0 \leq 1 - \kappa\hat{p}(T) \leq 1$, which eliminates the case when $x[i]$ could be negative and has to be set to 0.

Finally, from (4.26), we have the following upper bound for the rate function:

$$\inf_{0<T\leq T_I} \sum_{i=-T}^{-1} I\left(C + \frac{B}{T} - \Big(x^\star - \frac{w}{\hat{p}(T)}\Big)(1-\kappa\hat{p}(T))^{i+T+1} - \frac{w}{\hat{p}(T)}\right)$$

□

To understand the stability-elasticity trade-off, let us apply the marking elasticity parameter β to the upper bound on the rate function. Then, we observe that for a fixed T,

$$R(\beta) \triangleq C + \frac{B}{T} - \Big(x^\star - \frac{w}{\hat{p}_\beta(T)}\Big)(1 - k\hat{p}_\beta(T))^{i+T_l+1} - \frac{w}{\hat{p}_\beta(T)}$$

is increasing with respect to β, since $\hat{p}_\beta(T)$ is increasing with respect to β, for a fixed T. Thus, from Theorem 4.5.2, we observe that as we have an increasingly elastic marking function at the router, we get the larger rate function (and thus, a smaller queue overflow probability). However, from (4.19), increasing β causes the maximum allowable delay for stability to decrease. Thus, we have the stability-elasticity trade-off with the weighted proportional fairness, in a qualitatively similar form to that observed in (4.19) with the instant adaptation controller.

4.6 Numerical Results and Simulation

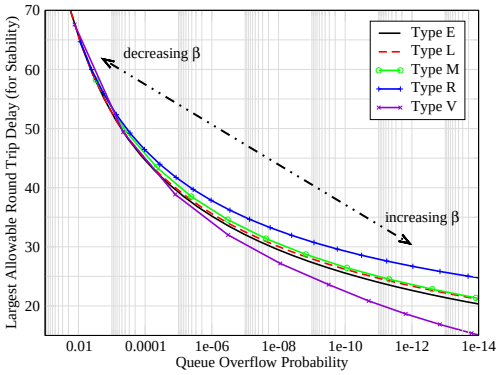
In this section, we present numerical examples under various environments to illustrate the trade-offs with both controllers, and present simulation results using the *ns-2* [33] packet simulator.

4.6.1 Instant Adaptation

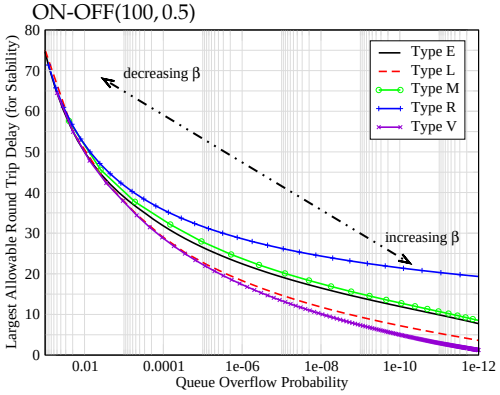
Stability-Elasticity Trade-off

We first illustrate the stability-elasticity trade-off with the instant adaptation algorithm in Figure 4.5. For each marking function in Table 4.1, we plot the trade-off between the largest allowable round-trip delay for stability and (the lower bound on) queue overflow probability (computed by the upper bound of the rate function in Theorem 4.4.2) as a parametric plot of β.

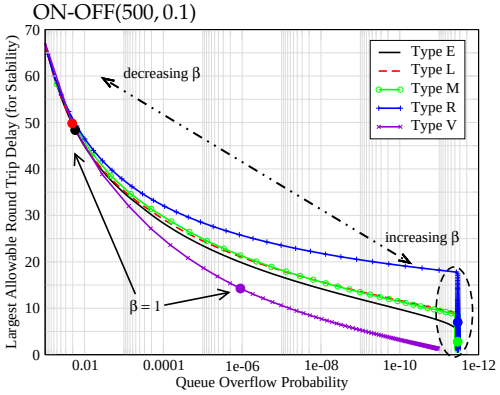
In Figure 4.5, each uncontrolled arrival process (from each real-time source) injects data at a mean rate (i.e., $y^\star$) of 50, the equilibrium rate ($x^\star$) from a controlled arrival process is set to be 45, and the per-flow link capacity (denoted by C) is 100 (thus, 95% link utilization



(a) $n = 100$, $w = 5$, $C = 100$, $\kappa = 0.2$ and ON-OFF(100, 0.5)

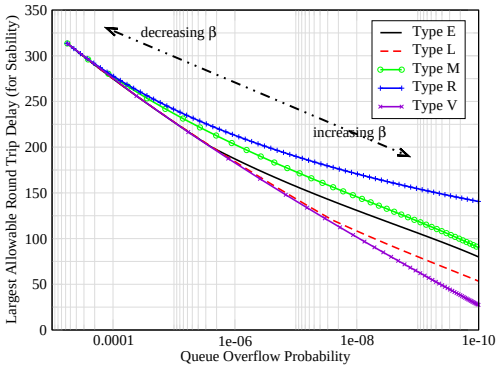


(b) $n = 500$, $w = 1$, $C = 100$, $\kappa = 0.9$ and ON-OFF(500, 0.1)

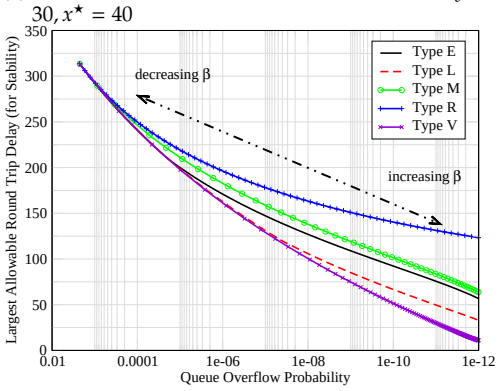


(c) $n = 500$, $w = 5$, $C = 100$, $\kappa = 0.2$ and ON-OFF(500, 0.1)

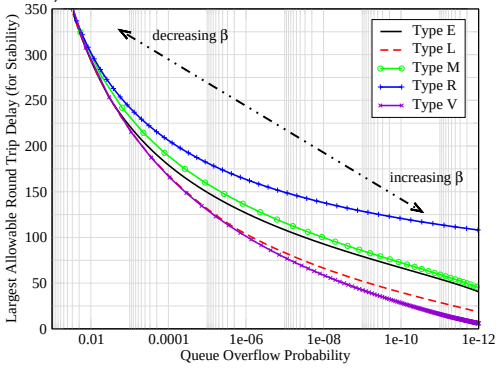
Figure 4.5: Stability-elasticity trade-off with ON-OFF (two state Markov) uncontrolled arrivals: ON-OFF(a, p) means that the ON rate is a with probability p and the OFF rate is 0 with probability $1 - p$.



(a) $n = 100$, $w = 1$, $C = 100$, $\kappa = 0.2$ and $z^\star = 70, y^\star = 30, x^\star = 40$



(b) $n = 100$, $w = 1$, $C = 100$, $\kappa = 0.2$ and $z^\star = 80, y^\star = 40, x^\star = 40$



(c) $n = 100$, $w = 1$, $C = 100$, $\kappa = 0.2$ and $z^\star = 90, y^\star = 40, x^\star = 50$

Figure 4.6: Stability-elasticity trade-off for different values of system equilibrium points ($z^\star$)

at the steady-state). We will present the results for different values of n, the number of flows in the network (thus, the total link capacity is $n \times C$). Under different values of w, κ,

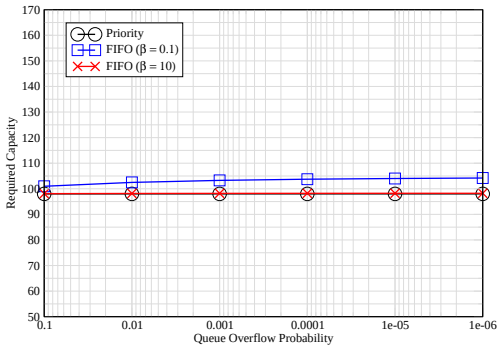
and the burstiness of uncontrolled flows, the plot clearly illustrates the trade-off between QoS for real-time flows and stability for controlled flows. The results also indicate that some marking functions may be "uniformly" better than others. In particular, among the marking functions that we have compared, our bounds indicate that a rate based version of REM (Type **R**) [10, 72] seems to provide the largest local-stability region for *any* given QoS requirement. An intuitive explanation for this is the following: From Theorem 4.4.2, it is clear that the QoS for the uncontrolled real-time flows with FIFO scheduling depends on the marking function behavior for arrival rates exceeding the per-flow link capacity C. In particular, the value of the rate function is proportional to the marking function value for arrival rates exceeding C (i.e., $I_F(\cdot) \propto p(z),\ z > C$). From Figure 4.4, we observe that among the example marking functions considered in Table 4.1 (which are normalized to have the same fixed point properties), the rate-based REM marking function seems to have the maximum slope for $z > z^\star$, which in turn implies a larger marking function value (as all the example marking functions have the same $p(z^\star)$). To analytically construct uniformly optimal marking functions is an interesting problem for future research.

In addition, we see different sensitivities to marking elasticity for different marking functions. The reason we have vertical lines in the rate based version of REM (Type **R**) and M/M/1 (Type **M**) marking function (in the dotted elliptical region in Figure 4.5-(d)) is that their original (non-warped) marking value $p(z)$ is 1, when $x > C$ (see Table 4.1). Thus, the queue overflow probability in this case decreases only until some threshold $\overline{\beta}$ and stays constant after this threshold.

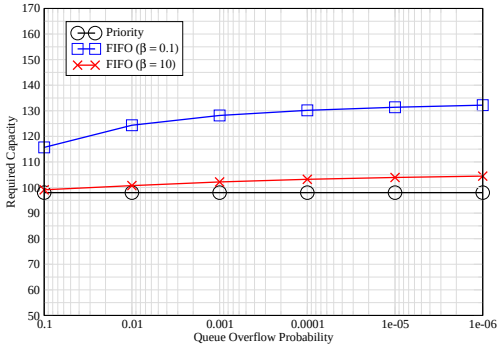
Scheduling-Elasticity Trade-off

To illustrate the scheduling-elasticity trade-off, Figure 4.7 and Figure 4.8 show the required per-flow link capacity with FIFO and priority scheduling to support a given QoS, for two values of the elasticity parameter β of type **V** marking function. The equilibrium rate from a controlled arrival process is set to be 48, and each uncontrolled arrival process (from each real-time source) injects data at a mean rate of 50. We use $w = 5$, and the number of flows (i.e., n) is 100.

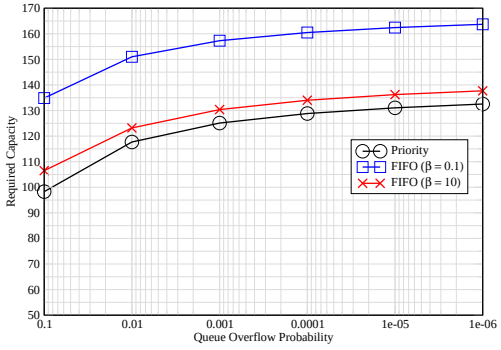
We first comment that the required link capacities for any QoS-requirement should



(a) ON-OFF(60, 5/6)

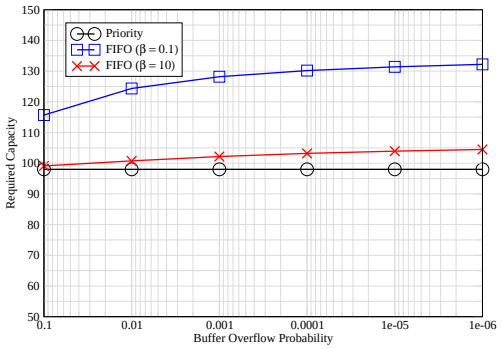


(b) ON-OFF(100, 0.5)

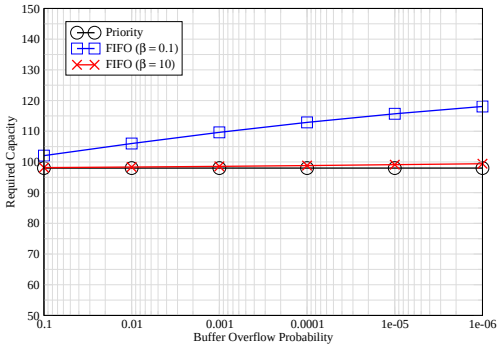


(c) ON-OFF(140, 5/14)

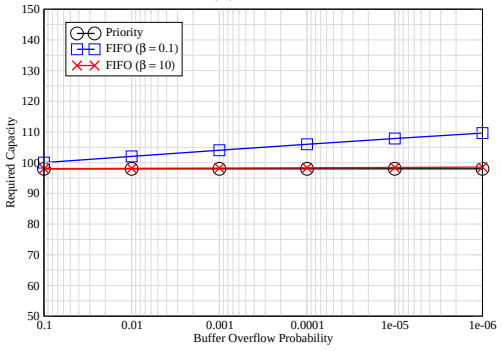
Figure 4.7: Scheduling-Elasticity Trade-off: Effect of Burstiness of Uncontrolled Flow. $w = 5$, $n = 100$, $x^\star = 48$, and $y^\star = 50$.



(a) $B = 1$



(b) $B = 5$



(c) $B = 10$

Figure 4.8: Scheduling-Elasticity Trade-off: Effect of Buffer Size, ON-OFF(100, 0.5) uncontrolled arrival. $w = 5$, $n = 100$, $x^\star = 48$, and $y^\star = 50$.

satisfy *queue stability condition*, i.e., the link capacities should be large enough such that the queue length is always finite. The sufficient condition for queue stability with both scheduling disciplines is to have the capacities should be larger than the equilibrium rate at the router queue, i.e., $C_P \geq x^\star + y^\star$ and $C_F \geq x^\star + y^\star$. Thus, Figure 4.7 and Figure 4.8 plots the maximum over the capacities governed by the queue overflow probability and

queue stability condition.

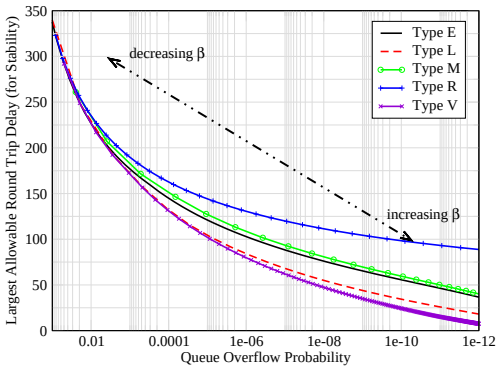
We observe that for a small value β, the difference between the capacities with FIFO and priority queueing is large for all values of the queue overflow probability. This is due to the fact that the controlled flows back-off sluggishly. On the other hand, for more elastic marking functions, the required capacities with both scheduling algorithms are very close.

For a less bursty uncontrolled arrivals (Figure 4.7-(a)), in priority scheduling, the queue stability condition (i.e., $C_P < z^\star = 98$) dominates the QoS condition (4.22), while for a more bursty arrivals (Figure 4.7-(c)), the QoS condition is stronger than the queue stability condition. In both cases, we observe that the required capacity with FIFO can be significantly decreased (to a value that is almost the same as that with priority scheduling) by increasing the marking elasticity.

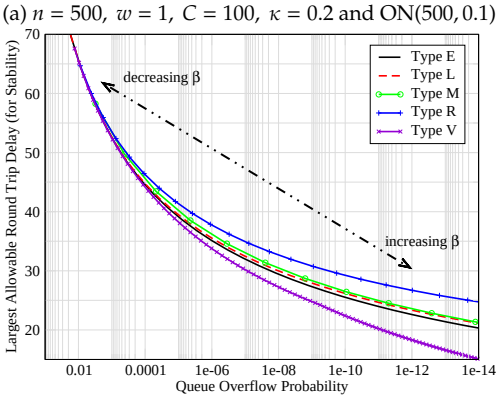
4.6.2 Weighted Proportional Fair Controller

Now, we study the trade-offs with the weighted proportional fair controller. First, as numerical examples, Figure 4.9 illustrates the stability-elasticity trade-off with the weighted proportional fair controller (with the same configuration parameters used as in Figure 4.5 with instant adaptation). We observe that analogous to the instant adaptation case, as the QoS parameter becomes more strict, the stability region is reduced. Also, importantly, we still observe that the type **R** marking function appears "uniformly better" than other marking functions, we have considered.

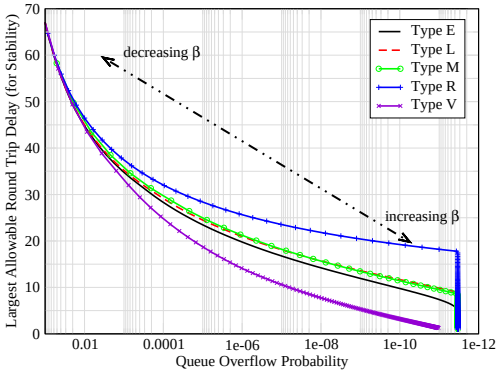
To study the trade-offs with the weighted proportional fair controller in a more practical scenario, we use the *ns-2* [33] packet simulator to validate our analysis. The network topology used in the simulation is same as that discussed in the analysis (see Figure 4.3). The number of uncontrolled and controlled flows (i.e., n) are set to be 100 throughout all the simulation results. The per-flow link capacity of the bottle-neck link is 100 pkts/sec (i.e., total capacity is 100×100 pkts/sec). The buffer size of the bottle-neck link is 100 pkts. We use the fixed size of packets (1000 bytes). Uncontrolled flows are modeled by discrete ON-OFF processes, where the burst-time and the idle-time are set to be 100 msec and 900 msec. The transmission rate in ON period is appropriately set such that the specified mean uncontrolled rate is achieved in different simulations. The parameter κ and w with the



(a) $n = 500,\ w = 1,\ C = 100,\ \kappa = 0.2$ and ON(500, 0.1)

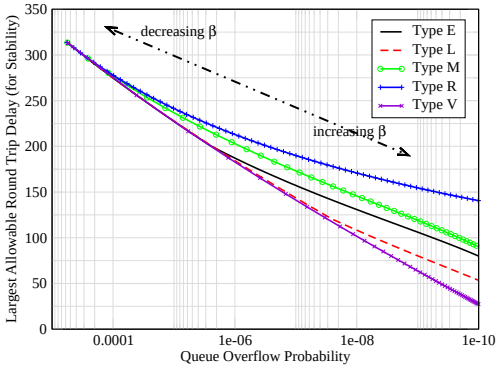


(b) $n = 100,\ w = 5,\ C = 100,\ \kappa = 0.2$ and ON(100, 0.5)

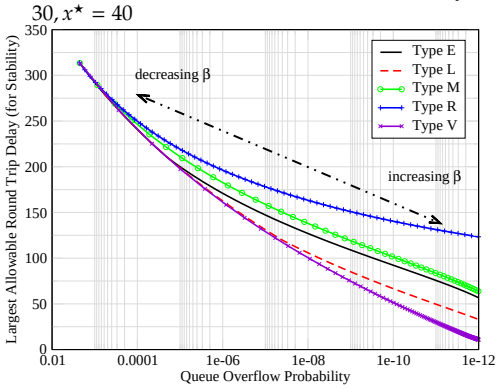


(c) $n = 500,\ w = 5,\ C = 100,\ \kappa = 0.2$ and ON(500, 0.1)

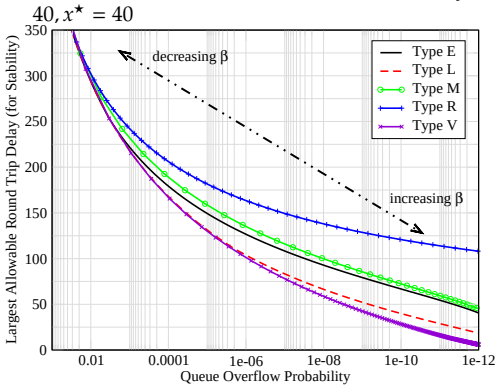
Figure 4.9: Stability-elasticity trade-off with Weighted Proportional Fair Controller



(a) $n = 100,\ w = 1,\ C = 100,\ \kappa = 0.2$ and $z^\star = 70, y^\star = 30, x^\star = 40$



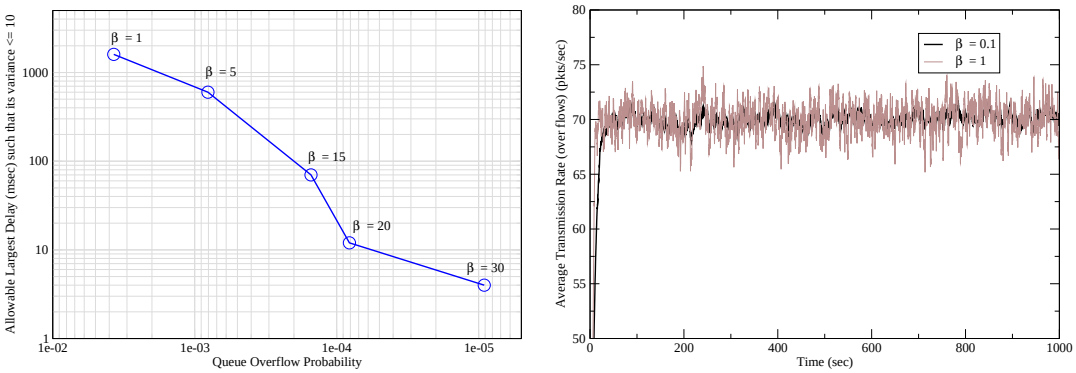
(b) $n = 100,\ w = 1,\ C = 100,\ \kappa = 0.2$ and $z^\star = 80, y^\star = 40, x^\star = 40$



(c) $n = 100,\ w = 1,\ C = 100,\ \kappa = 0.2$ and $z^\star = 90, y^\star = 40, x^\star = 50$

Figure 4.10: Stability-elasticity trade-off for different values of system equilibrium points ($z^\star$)

weighted proportional fair controller are set to be 1 and 5, respectively.



(a) Stability-elasticity trade-off (b) Trajectory of average transmission rates

Figure 4.11: Stability-elasticity trade-off: The trajectories in (b) show that for two values of β, the average remains the same. However, there is a trade-off between QoS and delay as observed in (a).

Table 4.3: Scheduling-Elasticity Trade-off: Required Per-flow Link Capacity for 10^{-5} queue overflow probability

	Priority	FIFO		
Marking Functions→	·	Type E	Type L	Type V
$\beta = 0.001$	95	119.2	119.5	118.7
$\beta = 1$	95	118.4	117.2	116.8
$\beta = 20$	95	110	110	108.5

Stability-Elasticity Trade-off

First, Figure 4.11 shows the stability-elasticity trade-off for the Type **V** marking function. The equilibrium rate ($x^\star$) of a controlled arrival process, the mean rate ($y^\star$) of an uncontrolled arrival process are set to be 70 pkts/sec and 20 pkts/sec, respectively. Figure 4.11-(b) shows the trajectories of average transmission rates with two different marking elasticities, from which we clearly observe that as we have larger β (more elastic marking function), the trajectory becomes more fluctuating, thus resulting in less stable behavior. For the plots in Figure 4.11-(a), we denote a flow to be stable if the transmission rate variance is less than 10 pkts/sec. In Figure 4.11-(a), we observe that the stability region becomes smaller, as we increase marking elasticity. However, increasing marking elasticity implies better QoS performance (lower queue overflow probability) with larger marking elasticity.

Scheduling-Elasticity Trade-off

Next, Table 4.3 shows the scheduling-elasticity trade-off with the weighted proportional fair controller. In this simulation, $(x^\star, y^\star)$ are set to be $(70, 25)$ pkts/sec, respectively. To see the scheduling-elasticity effect, for a fixed queue overflow probability (10^{-5}), we experimentally determine the required per-flow link capacity to satisfy the given queue overflow probability. With priority scheduling, it is observed that the queue stability condition is dominant; thus, the required capacities for different β values are all equal to 95 pkts/sec. With FIFO scheduling, we observe that the difference in the required link capacity between $\beta = 0.001$ and $\beta = 20$ is about 10 pkts/sec with all three types of marking functions. Further, the extra capacity that we need with FIFO scheduling to support the given QoS is shown to be about 15 pkts/sec. This validates the analytical result that the required link capacity decreases with increasing values of β.

Appendix

Proof of Lemma 4.4.1

Proof. First, we prove $h(z)$ is strictly increasing (thus, invertible) by showing that $h(z+\delta) - h(z) \geq \delta, \forall z \geq \underline{z}, \forall \delta > 0$. Since $p(z)$ is non-zero increasing when $z > \underline{z} > \underline{m}$, we have

$$\begin{aligned} h(z+\delta) - h(z) &= \delta + w\Big(\frac{1}{p(z)} - \frac{1}{p(z+\delta)}\Big) \\ &\geq \delta \end{aligned}$$

In addition, $h(z)$ is the sum of two concave functions since $-w/p(z)$ is concave from Assumption 4.3.2. Thus, $h(z)$ is a concave function.

Let us define $g \triangleq h^{-1} : \mathcal{R}^+ \cup \{0\} \to [\underline{z}, \infty)$. Then, we have

$$z[i] = g\Big(z[i] - \frac{w}{p(z[i])}\Big) = g(y[i])$$

Applying this to (4.5), we have

$$(g(y[i]) - y[i])p(g(y[i])) = w$$

The convexity of g follows immediately from the concavity of h. □

Proof of Theorem 4.4.1

Proof. We note that the authors in [71, 73] have proved that $Q(\vec{z}[-T_I, 0))$ is continuous with respect to $\vec{z}[-T_I, 0) \in \mathcal{R}^{T_I}$ in the topology endowed with uniform norm. Note that $\vec{z}[-T_I, 0)$ $(= \vec{y}[-T_I, 0) + \vec{x}[-T_I, 0))$ is the function (denoted by H) of $\vec{y}[-T_I, 0)$, since $\vec{x}[-T_I, 0)$ is determined by $\vec{y}[-T_I, 0)$ in the FIFO scheduling. Thus, from the definition of $\widetilde{Q}$, it suffices to show that the function $H : \mathcal{R}^{T_I} \mapsto \mathcal{R}^{T_I}$ is continuous (in the topology endowed with uniform norm) to prove that $\widetilde{Q}$ is continuous with respect to the uncontrolled arrival $\vec{y}[-T_I, 0)$. We will prove that for any given $\epsilon > 0$, and for two uncontrolled arrival processes, $\vec{y}[-T_I, 0)$ and $\vec{y}_\epsilon[-T_I, 0)$, such that $\|\vec{y}[-T_I, 0) - \vec{y}_\epsilon[-T_I, 0)\|_u < \epsilon$, there exists a function f such that $\|\vec{z}[-T_I, 0) - \vec{z}_\epsilon[-T_I, 0)\|_u < f(\epsilon)$, where $f(\epsilon) \overset{\epsilon \to 0}{\to} 0$.

First, note that at each time i, $z[i] (= x[i] + y[i])$ depends on only $y[i]$. Then, for a given $\epsilon > 0$, from the assumption that $\|\vec{y}[-T_I, 0) - \vec{y}_\epsilon[-T_I, 0)\|_u < \epsilon$ and the definition of uniform norm, we have

$$\sup_{0<T\le T_I} \left| \frac{\sum_{i=-T}^{i=1} (y[i] - y_\epsilon[i])}{T} \right| \quad < \quad \epsilon.$$

Then, the finiteness of T_I, there exists a finite non-negative constant K' (which is a function of T_I), such that $|y[i] - y_\epsilon[i]| < K'\epsilon$, for all $i \in [-T_I, \cdots, -1]$.

Now, observe that

$$\begin{aligned} \|\vec{z}[-T_I, 0) - \vec{z}_\epsilon[-T_I, 0)\|_u &= \sup_{0<T\le T_I} \left| \frac{\sum_{i=-T}^{i=-1} (z[i] - z_\epsilon[i])}{T} \right| \\ &\le \sum_{i=-T_I}^{i=-1} \Big| z[i] - z_\epsilon[i] \Big| = \sum_{i=-T_I}^{i=-1} \Big| g(y[i]) - g(y_\epsilon[i]) \Big| \\ &< T_I \max_{i=-T_I,\cdots,-1} \Big| g(y[i]) - g(y_\epsilon[i]) \Big|. \end{aligned} \tag{4.28}$$

Since $g(y)$ is continuous with respect to y from Lemma 4.4.1, (4.28) is arbitrarily small for an arbitrary small ϵ. This completes the proof. Then, the resulting rate function $I_F(B)$ immediately follows from [71, Theorem 9]. □

Proof of Lemma 4.4.2

Proof. Let us choose $\vec{y}^\star[-T^\star, 0) \in \mathcal{A}$. Suppose that $g(y^\star[k]) < C$, for a $k \in [-T^\star, 0)$. It suffices to show that we can find a new trajectory $\hat{\vec{y}}[-\hat{T}, 0)$ of no larger cost (rate function) than $\vec{y}^\star[-T^\star, 0)$ such that $g(\hat{y}[i]) \geq C,\ \forall i,\ -\hat{T} \leq i < 0$.

Define a new trajectory $\hat{\vec{y}}[-\hat{T}, 0),\ \hat{T} = T^\star - 1$ as follows.

$$\hat{y}[i] \quad = \quad \begin{cases} y^\star[i] & \text{if} \quad i \in [k+1, 0) \\ y^\star[i-1] & \text{if} \quad i \in [-T^\star + 1, k] \end{cases}$$

Then, we have constructed a new trajectory with cost as follows: $\sum_{-\hat{T}}^{-1} I(\hat{y}[i]) \leq \sum_{-T^\star}^{-1} I(y^\star[i])$, and $Q_F(\hat{\vec{y}}[-\hat{T}, 0)) \geq B$. In addition, if $g(y^\star[k]) < C$, for $k_1, k_2, \ldots \in [-T^\star, 0)$, we can inductively remove $k_1, k_2, \ldots$, and finally construct a new trajectory $\hat{\vec{y}}[-\hat{T}, 0) \in \hat{\mathcal{A}}$, with no larger cost. □

Proof of Lemma 4.4.3

Proof. It suffices to show that for any $\vec{a}[-T, 0) \in \mathcal{B}$, $\vec{a}[-T, 0) \in \hat{\mathcal{A}}$. We first have

$$\frac{1}{T}\sum_{i=-T}^{-1} a[i] \geq g^{-1}\Big(\frac{B}{T} + C\Big) \quad \Rightarrow \quad g\Big(\frac{1}{T}\sum_{i=-T}^{-1} a[i]\Big) \geq \frac{B}{T} + C$$

Also, since g is convex, we have

$$g\Big(\frac{1}{T}\sum_{i=-T}^{-1} a[i]\Big) \leq \frac{1}{T}\sum_{i=-T}^{-1} g(a[i])$$

from Jensen's inequality. This completes the proof. □

Proof of Lemma 4.4.4

Proof. First, it is clear that we get the optimum when $\sum_{i=-T}^{-1} z[i] = B + CT$, since f is increasing. Second, we claim that $C \leq z[i] \leq B + C,\ i \in J$, where $J = \{-T, -T+1, \ldots, -1\}$. Suppose that $z[j] > B + C$ for some $j \in J$. Then, We should have $z[k] < C$ for some $k \in J$, which contradicts the given condition.

Now, let $z^\star[-T] = B + C$, $z^\star[-T+1] = z^\star[-T+2] = \ldots = z^\star[-1] = C$. Since $C \leq z[i] \leq B + C$, $i \in J$, we can represent $z[i]$ by the following linear combination of $z^\star[i]$.

$$z[-i] \quad = \quad (1-\alpha_i)C + \alpha_i(B+C), \quad \forall i = 1, \ldots, T, \tag{4.29}$$

where $\sum_{i=-T}^{-1} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1$ for $i \in J$. From (4.29) and concavity of f,

$$\begin{aligned}
\frac{1}{T}\sum_{i=-T}^{-1} f(z[i]) &= \frac{1}{T}\sum_{i=-T}^{-1} f\Big((1-\alpha_i)C + \alpha_i(B+C)\Big) \\
&\geq f(C)\sum_{i=-T}^{-1}(1-\alpha_i) + f(B+C)\sum_{i=-T}^{-1}\alpha_i \\
&= (T-1)f(C) + f(B+C) = \frac{1}{T}\sum_{i=-T}^{-1} f(z^\star[i])
\end{aligned}$$

This completes the proof. □

Proof of Theorem 4.5.1

Proof. Similar to Theorem 4.4.1, $\vec{z}[-T_I, 0)$ is a function of $\vec{y}[-T_I, 0)$ (denoted by $\hat{H}$), and it suffices to show that the function $\hat{H} : \mathcal{R}^{T_I} \mapsto \mathcal{R}^{T_I}$ is continuous in the topology endowed with uniform norm. To do so, we prove that for any two uncontrolled arrival processes, $\vec{y}[-T_I, 0)$ and $\vec{y}_\epsilon[-T_I, 0)$, such that $\|\vec{y}[-T_I, 0) - \vec{y}_\epsilon[-T_I, 0)\|_u < \epsilon$, there exists a function f such that $\|\vec{z}[-T_I, 0) - \vec{z}_\epsilon[-T_I, 0)\|_u < f(\epsilon)$, where $f(\epsilon) \overset{\epsilon \to 0}{\to} 0$.

First, similar to the proof of Theorem 4.4.1, the assumption that $\|\vec{y}[-\infty, 0) - \vec{y}_\epsilon[-\infty, 0)\|_u < \epsilon$ implies that there exists a $K' > 0$ such that

$$|y[i] - y_\epsilon[i]| \quad < \quad K'\epsilon, \quad \forall i \in [-T_I, 0). \tag{4.30}$$

We now embed the discrete time trajectory of $\vec{y}[-T_I, 0)$ and $\vec{x}[-T_I, 0)$ in "continuous time," i.e., for $t \in \mathcal{Z}$, we let $x(t) = x[t]$, $y(t) = y[t]$ and use a straight-line approximation to interpolate between the times $t = i$, $i \in \mathcal{Z}$. Thus, we have the following differential equation

to represent the controlled flows dynamics in the continuous time:

$$\begin{aligned}\dot{x}(t) &= \kappa\Big(w - x(\lfloor t-d \rfloor)p\Big(x(\lfloor t-d \rfloor) + y(\lfloor t-d \rfloor)\Big)\Big)\\ \dot{x}_\epsilon(t) &= \kappa\Big(w - x_\epsilon(\lfloor t-d \rfloor)p\Big(x_\epsilon(\lfloor t-d \rfloor) + y_\epsilon(\lfloor t-d \rfloor)\Big)\Big),\end{aligned} \tag{4.31}$$

where $\sup_{-T_I \le t < 0} \delta(t) < K'\epsilon$.

Then, from [9, Lemma 3.2] and (4.31), we have

$$\begin{aligned}\sup_{-T_I \le i < 0} |x[i] - x_\epsilon[i]| &\le \sup_{-T_I \le t < 0} |x(t) - x_\epsilon(t)|\\ &\le 2LT_I e^{LT_I} K'\epsilon.\end{aligned} \tag{4.32}$$

Then, from (4.30) and (4.32), we have

$$\begin{aligned}\|\vec{z}[-T_I, 0) - \vec{z}_\epsilon[-T_I, 0)\|_u &= \sup_{0<T\le T_I} \left|\frac{\sum_{i=-T}^{i=-1}(z[i] - z_\epsilon[i])}{T}\right|\\ &\le \sup_{0<T\le T_I} \left|\frac{\sum_{i=-T}^{i=-1}(x[i] - x_\epsilon[i])}{T}\right| + \sup_{0<T\le T_I} \left|\frac{\sum_{i=-T}^{i=-1}(y[i] - y_\epsilon[i])}{T}\right|\\ &< 2LT_I e^{LT_I} K'\epsilon + K'\epsilon\end{aligned}$$

Recall that L is the Lipschitz parameter of the marking function. By letting $f(\epsilon) = 2LT_I e^{LT_I} K'\epsilon + K'\epsilon$, the result follows, since ϵ is arbitrary. □

Proof of Proposition 4.4.1

Proof. From Theorem 4.4.2, and the increasing property of $p_\beta(z)$ with respect to z, we have the following lower bound on $I_F(B)$:

$$\begin{aligned}I_F(B) &\ge \inf_{0<T\le T_I} TI\left(\frac{B}{T} + C_F(\beta) - \frac{w}{T}\Big(\frac{1}{p_\beta(B + C_F(\beta))} + \frac{(T-1)}{p_\beta(C_F(\beta))}\Big)\right)\\ &\ge \inf_{0<T\le T_I} TI\left(\frac{B}{T} + C_F(\beta) - \frac{w}{p_\beta(C_F(\beta))}\right)\end{aligned} \tag{4.33}$$

We let

$$\tilde{C}(\beta) = \frac{C_F(\beta) - w}{p_\beta(C_F(\beta))}, \quad \tilde{\delta} = \frac{\delta}{B}$$

Suppose that (4.22) is true, i.e., $\frac{\Lambda(\tilde{\delta})}{\tilde{\delta}} \leq \tilde{C}(\beta)$. Then, we have

$$\begin{aligned} \tilde{C}(\beta)\tilde{\delta} - \Lambda(\tilde{\delta}) \geq 0 \quad &\Rightarrow \quad (\tilde{C}(\beta) + x)\tilde{\delta} - \Lambda(\tilde{\delta}) \geq \tilde{\delta}x, \ \forall x > 0 \\ &\Rightarrow \quad \inf_{x \in \{B, B/2, \ldots\}} \frac{I(\tilde{C}(\beta) + x)}{x} \geq \tilde{\delta} \end{aligned} \tag{4.34}$$

Thus, we have $I_F(B) \geq \delta$ from (4.33) and (4.34). □

Chapter 5

A Hop-by-hop Congestion Control over a Wireless Multi-hop Network

5.1 Overview

This chapter focuses on congestion control over multi-hop, wireless networks. In a wireless network, an important constraint that arises is that due to the MAC (Media Access Control) layer. Many wireless MACs use a time-division strategy for channel access, where, at any point in space, the physical channel can be accessed by a single user at each instant of time.

In this chapter, we develop a fair hop-by-hop congestion control algorithm with the MAC constraint being imposed in the form of a channel access time constraint, using an optimization based framework. In the absence of delay, we show that this algorithm are globally stable using a Lyapunov function based approach. Next, in the presence of delay, we show that the hop-by-hop control algorithm has the property of spatial spreading. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. We derive bounds on the "peak load" at a node, both with hop-by-hop control, as well as with end-to-end control, show that significant gains are to be had with the hop-by-hop scheme, and validate the analytical results with simulation.

5.2 Introduction

We consider the problem of congestion control over wireless, multi-hop networks. Nodes in such networks are radio-equipped, and communicate by broadcasting over wireless links. Communication paths between nodes which are not in radio range of each other are established by intermediate nodes acting as relays to forward data toward the destination. The diverse applications of such networks range from community based roof-top networks to large-scale ad-hoc networks.

Over the past few years, the problem of congestion control has received wide-spread attention in the Internet context, where most of this research has focused on modeling, analysis, algorithm development of end-to-end control schemes (such as TCP), and adaptation of such schemes to ad-hoc networks [74]. Recent work on congestion control problem provides an optimization based framework for Internet congestion control and derives a differential equation based distributed solution in presence or absence of feedback delay with both primal [1–6, 6–8] and dual [10, 11] approach. Given routing path and bandwidth constraints, algorithms have been developed which converge and have a stable operation.

In a wireless context, however, an important additional resource constraint that arises is that due to the MAC (Media Access Control). To address this, we consider a wireless system, where multiple frequencies/codes are available for transmission, and enables parallel communication in a neighborhood using such orthogonal channels. The wireless MAC in such a system use a time-division strategy for channel access [75,76], where, at any point in space, *the physical channel can be accessed by a single user at each instant of time (see Section 5.3.1 for details).*

This chapter formulates an optimization framework for congestion control algorithm in wireless multi-hop networks with the constraint imposed by the MAC. We develop a distributed, *hop-by-hop* congestion control scheme, which is shown to be stable in the absence of propagation delays, and allocates bandwidth to various users in a proportionally-fair manner. In the presence of delay, we show that it has the property of *spatial spreading*. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. In Figure 5.1, we illustrate this effect. Consider a node accessed by a number

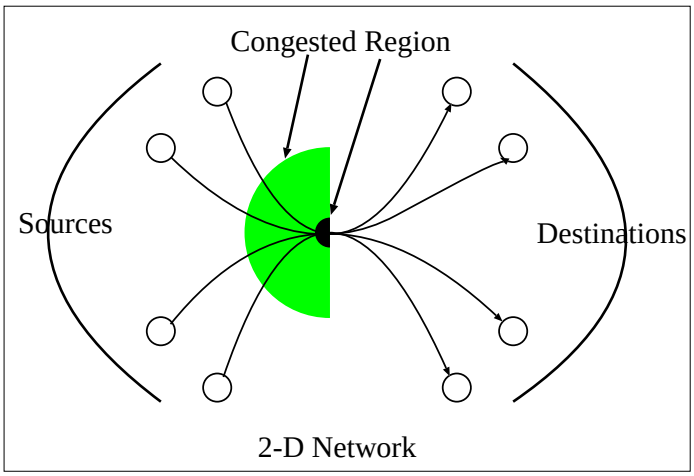


Figure 5.1: Spatial Spreading with hop-by-hop controllers

of flows. While an end-to-end control scheme could result in large transient overloads (due to delayed feedback) at a single node, a hop-by-hop scheme will "push-back" and cause congestion to occur over space, resulting in smaller peak overloads. Thus, even if the bottleneck node is very close to the receiver (the "worst-case" for a hop-by-hop scheme), there are potential gains to be had due to spatial spreading. *Hence, even if the total buffer requirement over the network is the same, the hop-by-hop scheme ensures that the buffers required are spatially spread.*

Hop-by-hop congestion control algorithms have been studied in the wire-line context [77–80]. Such schemes provide feedback about the congestion state at a node to the hop preceding it. The preceding node then adapts its transmission rate based on this feedback. Feedback is typically provided based on the queue length at the congested node. If the queue length exceeds a threshold, congestion is indicated and the preceding node is notified in order to decrease its transmission rate. It is well known that such schemes, by reacting to congestion faster than end-to-end schemes (the bottleneck node would send feedback *backward*, thus decreasing the delay in the control loop), result in better performance than a corresponding end-to-end scheme. However, Internet congestion control has been dominated by end-to-end schemes (in particular, TCP), and congestion control research in the recent past has focused on the end-to-end schemes, primarily due to scalability and deployability. Hop-by-hop schemes require to have per-flow state management

in intermediate nodes, which generates scalability problems.

However, in a wireless network, the number of flows per node is of a much smaller order than in the Internet. Further, wireless networks usually have per-flow queueing for reasons of packet scheduling [75, 81, 82], and the fact that different users are at different locations, thus requiring different physical layer strategies (such as the channel coding and modulation scheme of the power level). In fact, recent studies indicate that per-flow handling may be feasible even in the Internet. In the Internet context, the authors in [83] have measurements to suggest that even in the Internet, per-flow queuing is possible, as the number of *active* flows is in the tens or few hundreds, which can be supported. Thus, the hop-by-hop schemes seem feasible over a wireless multi-hop network.

Related work includes [84], where the authors consider max-min fair scheduling in the context of a wireless network using a similar model as that considered here for media access control (MAC). The authors develop a token based local scheduling policy at each node to ensure max-min fairness. The work on congestion (or rate) control algorithm in wireless ad-hoc networks also has been considered in the context of cross-layer design, where medium access control was jointly studied [85, 86]. In [85], the authors focus on proposing a unified framework for joint rate control and medium access scheduling using a dual approach, and proving stability the proposed algorithm in absence of delay. Recent work in [86] generalizes the resource constraint discussed in this chapter (by considering secondary contention as well as primary contention) imposed on MAC layer using flow contention graph, and proposes joint congestion control and media scheduling algorithm based on optimization framework in the context of only end-to-end controllers.

This chapter differs from the above-mentioned work in that under MAC constraints in (multi-channel) wireless ad-hoc networks, we develop rate based hop-by-hop as well as end-to-end congestion control algorithm with the objective of (weighted) proportionally-fair resource allocation among users. In particular, we prove that the proposed hop-by-hop algorithm is also stable in spite of coupling of transmission rates at each hop in absence of delay. Further, we derive bounds on peak queue lengths in the presence of propagation delay, both with an end-to-end and hop-by-hop scheme, and quantitatively demonstrate spatial spreading in the hop-by-hop control.

5.2.1 Main Contributions and Organization

The main contributions in this chapter are:

(i) We develop (weighted) proportionally-fair congestion control algorithms (both hop-by-hop as well as end-to-end) with the MAC constraint being imposed in the form of a channel access time constraint, using an optimization based framework. In the absence of delay, we show that these algorithms are globally stable using a Lyapunov function based approach. In particular, with the hop-by-hop algorithm, we have a collection of *coupled* controllers due to the fact that each node along a session's path implements a separate congestion controller. We show that this system of coupled controllers converges to a unique fixed point, which satisfies the proportional fairness condition.

(ii) We consider the evolution of these algorithms in the presence of propagation delay. We analytically show the effect of spatial spreading, by explicitly deriving the reduction in peak buffer overload under the hop-by-hop scheme for a general feed-forward network. We show that at a bottleneck node, the difference in the peak queue length between an end-to-end scheme and a hop-by-hop scheme is at least of order $L^{\alpha}N$, where L is the number of hops, N is the number of sessions, and for all $0 < \alpha < 1$.

We begin with a description of the system model in section 5.3, and discuss an utility function based network optimization framework. In section 5.4 and 5.5, we develop a distributed end-to-end and hop-by-hop congestion control algorithm, and prove the stability of both algorithms in absence of delay. Next, in section 5.6, we also develop a distributed end-to-end and hop-by-hop algorithm with delay, based on which, in section 5.7 we illustrate spatial spreading in a hop-by-hop algorithm by means of deriving bounds on the peak queue lengths in the presence of feedback delay. We provide simulation results in section 5.8 to validate the analysis.

5.3 System Model

5.3.1 Access Structure and Network Model

Consider a network with a set $\mathcal{L}$ of links, a set $\mathcal{V}$ of vertices (nodes), and let c_l be the finite capacity of link l, for $l \in \mathcal{L}$. Each vertex corresponds to a node in the network. Each data flow r in the network corresponds to an ordered sequence of links $l \in \mathcal{L}$, and we denote $\mathcal{R}$ as the set of possible sessions[1]. Thus, we model a wireless link between any two nodes in the network to have a finite positive capacity. We further assume that there is no data loss due to channel errors, and we ignore the effect of control data on the usage of channels, and implicitly assume no power control at the node.

In reality, wireless channels are time-varying [87], where each link has some average capacity which will depend on the physical layer scheme. However, in this chapter, we model the link to have a fixed capacity. Such a modeling is reasonable when the congestion controller changes *slowly* compared to a MAC packet transmission time, and MAC packet transmission spans *multiple channel states,* thus resulting in the channel capacity appearing constant (see also [84, 86] for a similar constant capacity model).

We next describe the access structure considered in this chapter. We consider a wireless system, where multiple frequencies/codes are available for transmission (using FDMA/CDMA), where frequency/code (resource) is allowed to be reused in the network, as long as the nodes using the same resource are sufficiently apart and are not involved in the same "contention," enabling parallel communications in a neighborhood using such orthogonal FDMA/CDMA channels (see [84, 88, 89] for additional discussion).

In this wireless system, a single transmission is intended for only one receiver, and each node has only a single transceiver, and hence only half-duplex communication is allowed. Further, a node can successfully receive from at most one other node at the same time. *Thus, at any instant of time, data flows that do not share nodes can transmit/receive simultaneously, but data flows that share a node cannot do so.* In other words, simultaneous transmissions can take place over links (i.e., between a pair nodes) as long as the links do not share a common node. We next describe the constraints on the data flows that follows from this wireless

[1] We use the words 'session' and 'flow' interchangeably throughout this chapter.

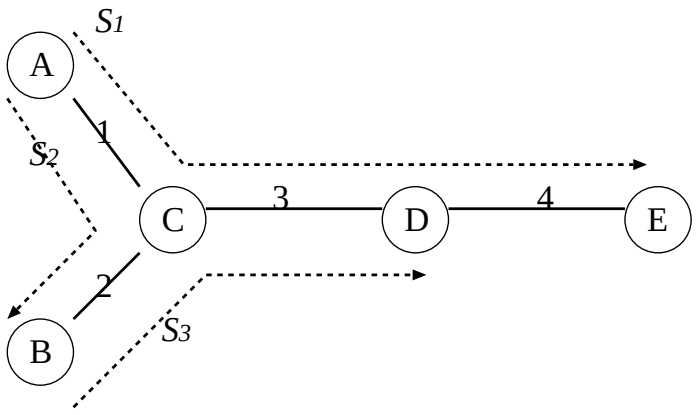


Figure 5.2: Example network for time and link constraint

system model.

5.3.2 Time Constraint

There are two types of constraints that are imposed, namely, *(i)* the link constraint and *(ii)* the time constraint. The *link constraint* (usually considered in wired networks) corresponds to the fact that the sum of date rates of all sessions that traverses through link $l \in \mathcal{L}$ is not greater than c_l, the capacity of link l. The *time constraint* means that at any instant of time, there can be only one instance of communication at a given node. To illustrate a fluid model for this constraint, we consider an example shown in Figure 5.2.

The network consists of three sessions S_1, S_2 and S_3, as shown in Figure 5.2. Let $x_i, i = 1, 2, 3$ be the data rate of the sessions respectively. We observe that the time constraint is imposed on each *node* in the network. Let us consider node 'C' in the figure, and define $y_{ij}, i \in \{1, 2, 3\}, j \in \{1, 2, 3, 4\}$ by

$$y_{ij} = \begin{cases} \frac{x_i}{c_j} & \text{if} \quad S_i \text{ traverses the link } j, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that y_{11} can be interpreted as the *fraction of time* node 'C' expends to receive data of session 1 from node 'A' over an *unit interval of time*. Similarly, y_{13} is interpreted as the fraction of time expended by node 'C' to transmit data of session 1 to node 'D' over an

unit interval of time. Similar interpretations hold for all y_{ij}. Thus, as total fraction of time expended at node 'C' cannot exceed 1, the *time constraint* at node 'C' is

$$\sum_{i\in\{1,2,3\},j\in\{1,2,3,4\}} y_{ij} \leq 1$$

Similar time constraints apply for all other nodes in the network.

In general, however, the time constraints presented above are *not sufficient* to ensure that a feasible MAC protocol exists [84, 89]. A feasible MAC always exists if the time constraints are relaxed by replacing the RHS of the expressions (i.e., the term '1') by a parameter $\rho \leq 2/3$ [89]. In this paper, we consider a fluid model for the MAC, and do not focus on the actual implementation (i.e., MAC protocol) of the resource sharing mechanism at each node. At the fluid time-scale, the details of these different MAC protocols manifest only as an *efficiency* factor that is captured by the parameter $\epsilon_j,\ j \in \mathcal{V}$, which governs the fraction of time that the time resource at each node can be used for successful data transfer. For example, the time constraint at node 'C' is represented by:

$$\sum_{i\in\{1,2,3\},j\in\{1,2,3,4\}} y_{ij} \leq 1-\epsilon_3$$

The efficiency factor is chosen such that some MAC protocol is feasible for the given network topology and session configuration. From our earlier discussion, $\epsilon_j \geq 1/3,\ j \in \mathcal{V}$ ensures that a time-division MAC is *always feasible independent of the network topology*. Further, an inefficient MAC scheme would be associated with a larger value of $\epsilon_j,\ j \in \mathcal{V}$. For example, an ideal MAC algorithm would allow the maximum possible (subject to MAC feasibility) time-resources at each node to be used for successful data transfer. However, an ALOHA based MAC would have inefficiencies associated with it, which would allow only a fraction of the time resources at a node to be used for successful data transfer.

Table 5.1 presents the link and time constraints for the network in Figure 5.2. As we can observe from the table, the link constraints are *subsumed* by the time constraints. Any link constraint is trivially a time constraint, if it is the only flow and terminates at the node. In all other cases, the time constraint is strictly stronger than a link constraint. Thus, we

Table 5.1: Link and time constraints for the example network in Figure 5.2

Link	Link Constraint	Node	Time Constraint
1	$\frac{x_1+x_2}{c_1} \leq 1$	1	$\frac{x_1+x_2}{c_1} \leq 1-\epsilon_1$
2	$\frac{x_2+x_3}{c_2} \leq 1$	2	$\frac{x_2+x_3}{c_2} \leq 1-\epsilon_2$
3	$\frac{x_1+x_3}{c_3} \leq 1$	3	$\frac{x_1+x_2}{c_1} + \frac{x_2+x_3}{c_2} + \frac{x_1+x_3}{c_3} \leq 1-\epsilon_3$
4	$\frac{x_1}{c_4} \leq 1$	4	$\frac{x_1+x_3}{c_3} + \frac{x_1}{c_4} \leq 1-\epsilon_4$
		5	$\frac{x_1}{c_4} \leq 1-\epsilon_5$

do not need to consider link constraints, and will henceforth restrict ourselves to only time constraints.

5.3.3 An Optimization Problem

Let us denote $N(\mathcal{L})$, $N(\mathcal{V})$, and $N(\mathcal{R})$ as the number of links, nodes, and sessions, respectively. For any link l and session r, let $A_{lr} = \frac{1}{c_l}$ if link l is in the path of flow r, and 0 other-wise. Thus, we define the matrix $A \in \mathcal{R}^{N(\mathcal{L})\times N(\mathcal{R})}$ by

$$A = \begin{cases} A_{lr} = 1/c_l & \text{if link } l \text{ in session } r, \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Similarly, define $G_{vl} = 1$ if link l is incident on node v, and 0 other-wise. Thus, define the matrix $G \in \{0,1\}^{N(\mathcal{V})\times N(\mathcal{L})}$ is defined by

$$G = \begin{cases} G_{vl} = 1 & \text{if link } l \text{ incident on node } v, \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Using G and A, time constraint for a given network can be expressed as:

$$GA\boldsymbol{x} \leq 1-\epsilon, \tag{5.3}$$

where $\epsilon = (\epsilon_j \in [0,1])$, $j \in \mathcal{V}$, and $\boldsymbol{x}$ corresponds to the vector of user data rates.

For each user (session) r, let x_r be the data transmission rate. Associated with each

user (session) is a utility function $U_r(\cdot)$, which is the "reward" or utility that user r gets by transmitting at the rate of x_r (see [1] for further discussion). Assume that the utility $U_r(x_r)$ is an increasing, strictly concave, and continuously differentiable function of x_r over the range $x_r \geq 0$. In this chapter, we restrict ourselves to weighted proportionally fair utility functions of the form $U_r(\cdot) = w_r \log(\cdot)$. From a resource allocation point of view, the resource allocation achieved under any concave and increasing utility functions can be achieved by a weighted proportionally-fair allocation[2] [90] through appropriate choice of weights $\{w_r\}$.

The objective is to maximize total utility in the network subject to the link and time constraints. In this chapter, we develop congestion control mechanisms to share the time resources in the network in a (weighted) proportionally fair manner. Thus, with session rates $\boldsymbol{x} = (x_r, r \in R)$, we need to solve

$$\mathbf{P} : \max \sum_{r \in R} w_r \log(x_r)$$

subject to

$$\begin{aligned} GA\boldsymbol{x} &\leq 1 - \epsilon, \\ \boldsymbol{x} &\geq 0 \end{aligned}$$

As the cost function is strictly concave and the constraint set is convex, there is a unique solution to **P**. In following sections, we develop a decentralized congestion control algorithms (both hop-by-hop and end-to-end) to address **P**.

5.4 Distributed end-to-end Algorithm

5.4.1 Algorithm Description

In this section, we develop an distributed end-to-end congestion control algorithm to solve **P**. In this chapter, we do not consider the optimal global MAC scheduling problem. Thus, "distributed" in this paper means that the congestion controller itself is distributed, since each source needs only local information on resource usage and feedback from the network

[2]However, the transient dynamics of a decentralized controller may be different.

to adjust its transmission rate. Similar discussion holds for the hop-by-hop congestion control presented in Section 5.5.

As the optimization problem has a strictly concave cost function, and convex constraints, we solve **P** using Lagrange multipliers. The Lagrangian of the problem **P** is:

$$L(\boldsymbol{x}, \lambda) \quad = \quad \sum_{r \in R} w_r \log(x_r) - \lambda^T \Big(GAx - (1 - \epsilon) \Big). \tag{5.4}$$

We denote the input and output link of a session r on the node v when a session r goes through v as $l_i(v, r)$ and $l_o(v, r)$, respectively (for instance, in Figure 5.2, are $l_i(3, 1) = 1$ and $l_o(3, 1) = 3$). For completeness, for the source and destination nodes, we define $c_{l_i(s(r),r)} = \infty$ and $c_{l_o(d(r),r)} = \infty$ respectively, where the source and the destination node of session r are denoted by $s(r)$ and $d(r)$. Let us denote $A_j(r)$ as the set of all downstream nodes from node j in the path of session r. Thus, $A_{s(r)}(r)$ is the collection of all nodes in the path of session r.

By differentiating (5.4), we have

$$\frac{dL}{dx_r} = \frac{w_r}{x_r} - \sum_{j \in A_{s(r)}(r)} \Big(\Big(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}} \Big) \lambda_j \Big) = 0 \tag{5.5}$$

Therefore, the unique solution to the problem **P** is given by the following condition:

$$x_r \quad = \quad \frac{w_r}{\sum_{j \in A_{s(r)}(r)} \Big(\Big(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}} \Big) \lambda_j \Big)} \tag{5.6}$$

We now present rate adaptation mechanisms for session sources. At each time t, we denote the transmission rate of session r by $x_r(t)$. Suppose that $x_r(t)$ adapts according to

$$\dot{x_r}(t) = \kappa \Big(w_r - x_r(t) \sum_{j \in A_{s(r)}(r)} \Big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}}) \lambda_j(t) \Big) \Big), \tag{5.7}$$

where

$$\lambda_j(t) \quad = \quad \beta p_j \Big(\sum_{s \in D(j)} x_s(t) \Big(\frac{1}{c_{l_i(j,s)}} + \frac{1}{c_{l_o(j,s)}} \Big) \Big), \tag{5.8}$$

$p_j(y)$ is a *marking function* at node j, and determines the fraction of flow to be marked. Here,

$D(j)$ corresponds to the set of sessions incident on node j, and $\beta > 0$ is a fixed constant.

This function is an indicator of (time) congestion at a node, and sources adapt based on the congestion indication [1, 2]. As in the Internet context, this function is assumed to be a continuous, increasing function with range $[0, 1]$.

Observe that (5.7) is analogous to the differential equation developed in [1]. However, (5.7) differs from the algorithm of [1] in that (5.7) handles relative transmission or reception times instead of actual rates. In practice, the incoming/outgoing rates could be known to the nodes by measuring the amount of data over a small interval of time.

To understand the intuition for (5.7), observe that λ_j is interpreted as the price for using node j per unit time. In addition, $x_r(t)(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})$ is the fraction of time the MAC at node j expends in receiving and re-transmitting (to the next hop) the data from session r. As *time* is the resource in our formulation, the total cost of using node j equals $x_r(t)(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j$. Thus, the source congestion control mechanism tries to equalize the aggregate cost $x_r(t) \sum_{j \in A_{s(r)}(r)} \Big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j(t) \Big)$ with w_r.

5.4.2 Marking function

Corresponding to each node j in the network is a marking function $p_j(\cdot)$. In this chapter, we consider a marking function of the form (at node j)

$$p_j(y) \quad = \quad \Big(\frac{y - \tilde{t}_j}{y} \Big)^+. \tag{5.9}$$

This marking function has the interpretation of the fraction of time lost when the time usage at node j exceeds a certain level $\tilde{t}_j$, called the "virtual time" (This is similar to the concept of "virtual capacity" in [2]). As seen in (5.8), the parameter y of $p_j(y)$ is the sum of MAC time utilization by *all flows,* both incoming and outgoing, at node j.

Thus, as discussed earlier, $\beta(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})p_j(y)$ marks the fraction of flow which exceeds a *time threshold* $\tilde{t}_j$. Observe that the total time utilization at the MAC cannot exceed 1. Thus, $\tilde{t}_j < 1$ is a parameter that *controls the desired time utilization* at the node. For instance, for an inefficient MAC (say, random access), one would set $\tilde{t}_j \ll 1$. If the MAC is more inefficient at node j, the (equilibrium) utilization at that node is smaller. This means that the node

has to mark more packets. Thus, an inefficient MAC has low value of $\tilde{t}_j$.

We will discuss the choice of this parameter in Section 5.4.3.

5.4.3 Stability Analysis

In this section, we show that the system of controllers defined in (5.7) is globally stable. The proof is analogous to that in [1]. Let the function $W(\boldsymbol{x})$ be defined by

$$W(\boldsymbol{x}) = \sum_{r \in \mathcal{R}} w_r \log x_r - \beta \sum_{j \in \mathcal{V}} \int_0^{\sum_{s \in D(j)} x_s(t)\left(\frac{1}{c_{l_j(j,s)}} + \frac{1}{c_{l_o(j,s)}}\right)} p_j(y)\, dy \tag{5.10}$$

We can show that $W(\cdot)$ is a strictly concave function, with a unique equilibrium $\boldsymbol{x}^*$.

Then, with a slight extension to [91], we can find virtual times $\{\tilde{t}_j\}$ at each node $j \in \mathcal{V}$, and β, such that the unique maximum of the optimization problem given by (5.10) also solves **P**. Thus, the equilibrium rate $\boldsymbol{x}^*$ can be suitably chosen by choosing appropriate values for the time thresholds $\{\tilde{t}_j\}$, and the constant β. In particular, $\{\tilde{t}_j\}$ is chosen such that the equilibrium $\boldsymbol{x}^*$ solves the optimization problem **P** discussed in Section 5.3.3. Due to space limitation, we skip the proof. Adaptively choosing these parameters in a manner similar to that in [27, 91] is a topic for future research. We now show that the congestion controllers described by (5.7) and (5.8) converge to this equilibrium point. Now, we have the following proposition on the stability of the end-to-end congestion controller.

Proposition 5.4.1. *$W(\boldsymbol{x})$ is a strictly concave, Lyapunov function for the system of differential equations (5.7). The unique value of $\boldsymbol{x}$ maximizing $W(\boldsymbol{x})$, denoted by $\boldsymbol{x}^*$ is a stable point of the system, to which all trajectories converge.*

Proof. The proof is analogous to that in [1], where we use the Lyapunov function defined in (5.10). We skip the details. □

5.5 Distributed hop-by-hop Algorithm

In this section, we develop a distributed hop-by-hop algorithm for congestion control. First, we observe that the congestion controller at the source of each session reacts based on the

sum of the congestion prices at each node. Instead of passing this feedback downstream as in the end-to-end algorithm, one could envisage a scheme where each node passes the (partial sum) price upstream. In other words, each node adds its current congestion cost to that it received from a downstream node, and passes this information toward the upstream node. The source will ultimately receive the sum of all price information from the corresponding downstream nodes and use the information for controlling rates. We refer to Figure 5.3 for the illustration of the hop-by-hop algorithm.

$$\text{feedback A: } \tfrac{1}{c_1}\lambda_A(t) + \big(\tfrac{1}{c_1} + \tfrac{1}{c_2}\big)\lambda_B(t) + \tfrac{1}{c_2}\lambda_C(t)$$
$$\text{feedback B: } \big(\tfrac{1}{c_1} + \tfrac{1}{c_2}\big)\lambda_B(t) + \tfrac{1}{c_2}\lambda_C(t)$$
$$\text{feedback C: } \tfrac{1}{c_2}\lambda_C(t)$$

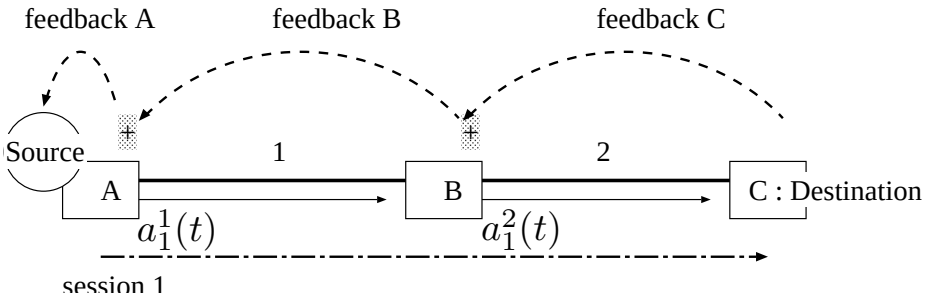


Figure 5.3: hop-by-hop Congestion Control Algorithm

The basic idea of a hop-by-hop algorithm is that every node in the path of the session operates a congestion control algorithm. In Figure 5.3, the congestion price at node *C* is passed to the upstream node *B*. Node *B* computes its local congestion price and adds it to the congestion price from node *C*. Node *B* adapts its transmission rate to node *C* based on this sum of congestion prices. In addition, node *B* passes this sum of two prices to the upstream node *A*. Using this "price passing" method, the source of session 1 receives aggregate congestion price from its downstream nodes and controls its transmission rate based on it.

Let us denote $a_r^i(t)$ as the *actual transmission rate* at the *i*-th hop of session *r* in the hop-by-hop control algorithm. Corresponding to each node *i* along the path of session *r*, is

a *virtual transmission rate* $c_r^i(t)$, which is described by

$$\dot{c}_r^i(t) = \kappa\Big(w_r - a_r^i(t) \sum_{j \in A_k(r)} \left(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}}\right) \lambda_j(t)\Big), \tag{5.11}$$

$$a_r^i(t) = \min[c_r^i(t), a_r^{i-1}(t)], \tag{5.12}$$

where k is the node corresponding to the i-th hop of session r. $\{\lambda_j(t)\}$ are defined similar to that in (5.8), but with the actual transmission rates instead of the source transmission rates. Along the path of each flow r, and for each hop i, the initial conditions for the virtual transmission rates are assumed to satisfy $c_r^i(0) \geq c_r^{i-1}(0)$ (in particular, all of them could be equal).

Thus, in the above algorithm, we sum over all prices downstream along session r. Thus, each node operates a (per-flow) controller based on the perceived congestion due to *downstream nodes*, and determines the maximum rate it can transmit at (the virtual transmission rate). The actual rate it chooses transmits at the rate of the minimum of the *incoming* data rate[3] from $i-1$-th hop node in the session's path (the previous hop node), i.e, $a_r^{i-1}(t)$, and the maximum possible rate $c_r^i(t)$.

We comment that at each intermediate node, the controller has knowledge of the local link rates, as well as the "ramp-up" constant w_r for each of the sessions that is incident on the node. It can be shown that the stability analysis and later analysis are valid even if the node uses an upper bound on the ramp-up constant. Thus, from an implementation perspective, one could assume that $\{w_r\}$ are globally bounded by some value w, and use this value at each intermediate node. Heuristically, the convergence proofs are valid even when a bound is used because the data transmission rate into the network is ultimately governed by the source, which will use the correct value of w_r. However, to keep the exposition simple, we will use the exact value of w_r at each node in this chapter.

Proposition 5.5.1. *The transmission rates for the hop-by-hop controller described in (5.11) and (5.12) converge to the equilibrium value* $\boldsymbol{x}^* = (x_1^*, \ldots, x_{N(\mathcal{R})}^*)^T$ *given in Proposition 5.4.1. In particular, for each route r, and for each hop i,* $a_r^i(t) \to x_r^*$ *as* $t \to \infty$.

[3]For the source node for each flow, (5.12) is not considered, as there is no upstream node. Instead we let the actual and virtual transmission rates to be the same.

Proof. At any fixed time t, note that $a_r^i(t)$ *decreases* along a flow path. This follows from (5.12), where the min implies that $a_r^i(t) \leq a_r^{i-1}(t)$. Further, from (5.11), it follows that the the sum of the Lagrange multipliers *decreases* along a flow path (as smaller number of non-negative terms added as we get closer to the destination). These two facts imply that

$$\frac{d}{dt}\left(c_r^i(t) - c_r^{i-1}(t)\right) \quad \geq \quad 0,$$

from (5.11). This observation, along with the ordering of the initial conditions of $\{c_r^i(0)\}$ implies that $c_r^i(t)$ *increases* along a flow path (i.e., for increasing values of i). In other words, for each i, r, t, we have

$$c_r^i(t) \quad \geq \quad c_r^{i-1}(t) \tag{5.13}$$

Next, for each i, r, from (5.12), we have that

$$a_r^i(t) \quad \leq \quad c_r^i(t) \tag{5.14}$$

Thus, from (5.13) and (5.14), we have $c_r^i(t) \geq c_r^{i-1}(t) \geq a_r^{i-1}(t)$. Hence, it follows that

$$a_r^i(t) \quad = \quad \min\{c_r^i(t), a_r^{i-1}(t)\} = a_r^{i-1}(t).$$

This implies that, for each flow r, the actual transmission rates $\{a_r^i(t), i = 1, 2, \ldots\}$ are the same over all hops. This in-turn implies that the source dynamics for flow r is the same as source dynamics of the end-to-end controller. Hence, from Proposition 5.4.1, the result follows. □

We finally remark that in practice, $c_r^i(t)$ will be bounded by a finite constant (typically, the output link capacity). This is to ensure that in the absence of congestion, the virtual capacity does not become unbounded (this can happen for instance, if the congestion occurs only upstream to a particular node, and thus, all downstream controllers are redundant).

5.6 Congestion Control with Delay

In the previous section where we proved stability, we assumed that the time resource was large enough so that queueing did not occur (or equivalently, the time threshold $\tilde{t}$ are suitably chosen). In this section, we do not make such an assumption. We will study the dynamics with queueing in the presence of feedback delay.

For the end-to-end algorithm, we denote the output transmission rate of session r traversing the link l by $x_{r,l}(t)$. Thus, $x_{r,l_i(j,r)}(t)$ and $x_{r,l_o(j,r)}(t)$ are the incoming input and outgoing transmission rate of session r at node j in the end-to-end algorithm, respectively. Similarly, for the hop-by-hop algorithm, we denote the actual and maximum (virtual) sending rate of session r traversing the link l by $a_{r,l}(t)$ and $c_{r,l}(t)$, respectively. Thus, $a_{r,l_i(j,r)}(t)$ and $a_{r,l_o(j,r)}(t)$ are the actual incoming input and actual outgoing transmission rates of session r at node j respectively.

Finally, each node has a per-flow buffer to temporarily store data before forwarding. We denote the queue length of session r at node j by $q_{rj}(t)$.

5.6.1 The End-to-End Controller with Delay

Unlike in the delay-free case considered in Section 5.4, queueing can occur at intermediate nodes due to feedback delay. In this section, we describe the detailed dynamics of rates for a session at each node.

For each node j, let us define $E_I^j(t)$ by

$$E_I^j(t) \quad = \quad \sum_{r \in D(j)} \frac{x_{r,l_i(j,r)}(t)}{c_{l_i(j,r)}}$$

Thus, $E_I^j(t)$ is the fraction of the time resource at the MAC of node j consumed by incoming flows, and $D(j)$ corresponds to the set of sessions incident on node j. We will assume that the MAC protocol at the nodes ensure that $E_I^j(t) < 1$. Thus, if a timing overload occurs at a node, data loss will occur, causing unsuccessful transmissions to be queued at the preceding hop (where the data was transmitted from). We assume a suitable error and collision detection mechanism exists such that data is queued in case of timing overload.

Thus, from a fluid model perspective, we can assume that the successful data transmission into a node j satisfies $E_I^j(t) < 1$. In addition, a poor MAC protocol may not be able to support a time utilization of '1' (for instance an ALOHA based MAC would have a maximum time utilization less than 0.36). However, in the following discussions, we will assume that the MAC can support a time utilization of '1' for notational ease. The results that are presented can be easily generalized to non-ideal MACs by suitable scaling. Let us now define

$$E_O^j(t) = \sum_{r \in D(j)} \frac{x_{r,l_i(j,r)}(t)}{c_{l_o(j,r)}}$$

The interpretation of $E_O^j(t)$ is the following: If there is no congestion at the node j, the output transmission rates would simply be equal to the incoming rate. $E_O^j(t)$ is the time utilization at the MAC in such a case.

We now consider the following two cases.

(i) $E_I^j(t) + E_O^j(t) > 1$

As the time utilization at the node will exceed '1' if the output flow rates equal the input flow rates, we decrease the transmitted output rates such that the time constraint is met. In other words, we choose $\alpha(t) \in (0, 1]$ such that $E_I^j(t) + \alpha(t)E_O^j(t) = 1$, and set the output transmission rate by $x_{r,l_o(j,r)}(t) = \alpha(t)x_{r,l_i(j,r)}(t)$. The remaining flow (of fraction $1 - \alpha(t)$) is queued at node j.

(ii) $E_I(t) + E_O(t) \leq 1$

In this case, the output flow rate for each session can be set to *at least* the input rate of the corresponding session. If some of the sessions have strictly positive queue lengths, i.e., users with backlogged queues (corresponding to congestion in the past), these users are allocated output rates that are greater than their input rates. The rates will be allocated in some fair manner (for example, a proportional rate increase to all backlogged users), subject to the timing constrain being met. Let us denote $Q_j^+(t)$ be the set of backlogged sessions at node j at time t. We choose $\alpha(t) > 1$ such that the time utilization at the node is less than or equal to one, and for all sessions $r \in Q_j^+(t)$, $x_{r,l_o(j,r)}(t) = \alpha(t)x_{r,l_i(j,r)}(t)$.

5.6.2 The Hop-by-Hop Controller with Delay

We now develop the dynamics of the hop-by-hop controller with delay. As in the Section 5.6.1, we define the total time utilization due to incoming flows at node j, by

$$H_I^j(t) = \sum_{r \in D(j)} \frac{a_{r,l_i(j,r)}(t)}{c_{l_i(j,r)}}$$

Let us denote $Q_j^+(t)$ be the set of backlogged sessions at node j at time t, and define

$$H_O^j(t) = \sum_{r \in D(j) r \in Q_j^+(t)} \frac{c_{r,l_o(j,r)}(t)}{c_{l_o(j,r)}} + \sum_{\substack{r \in D(j) \\ r \notin Q_j^+(t)}} \frac{\min[c_{r,l_o(j,r)}(t), a_{r,l_i(j,r)}(t)]}{c_{l_o(j,r)}}$$

where $c_{r,l_o(j,r)}(t)$ is the maximum possible rate for flow r at node k, and is described by (5.11). We now consider two cases:

(i) $H_I^j(t) + H_O^j(t) \leq 1$

In this case, there is no scarce time resource at this node. If the user queues are zero, the output rate is simply equal to the input rate. In general, the output rate for session r is given by

$$a_{r,l_o(j,r)}(t) = \begin{cases} \min[c_{r,l_o(j,r)}(t), a_{r,l_i(j,r)}(t)] & \text{if} \quad q_{rj}(t) = 0, \\ c_{r,l_o(j,r)}(t) & \text{if} \quad q_{rj}(t) > 0 \end{cases}$$

(ii) $H_I^j(t) + H_O^j(t) > 1$

In this case, the time resource at node j is potentially not sufficient to handle the output rate. Similar to Case (i) for the end-to-end controller in Section 5.6.1, we choose $\alpha(t) \in [0, 1)$ such that $H_I^j(t) + \alpha(t) H_O^j(t) = 1$, and set the output transmission rate correspondingly.

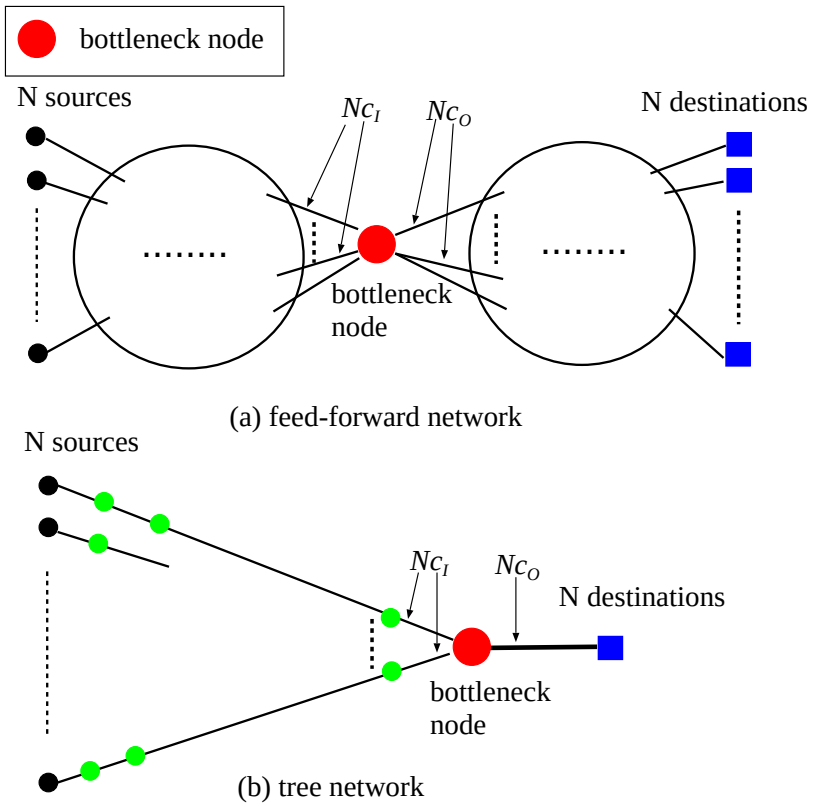


Figure 5.4: Networks with Spatial Spreading: one hop rtt: d_H, and end-to-end rtt of each source: d_R

5.7 Spatial Spreading

In this section, we derive the peak occupied buffer size with the end-to-end controller as well as with the hop-by-hop controller described in Section 5.6. We consider the evolution of these algorithms in the presence of propagation delay. We analytically show the effect of spatial spreading by explicitly deriving the reduction in peak buffer overload under the hop-by-hop scheme.

Consider a feed-forward network (Figure 5.4-(a)) with N sessions, where the links are symmetric, and have the same propagation delay. Let us denote the link (one-hop) round trip delay by d_H (i.e., a link propagation delay = $d_H/2$). Further, we assume that all the sessions has the same end-to-end round trip delay (denoted by d_R). Thus, all sessions have $L = d_R/d_H$ number of hops in their end-to-end path. We assume that a single bottle-neck

in the network (see Figure 5.4), and that the delays from the N sources to the bottleneck are all equivalent. We also assume that the capacities of other links than those of input or output links of the bottle-neck node are well provisioned, such that congestion occurs only at the common point for all the flows (the bottleneck node in Figure 5.4-(a)). We will first consider the end-to-end scheme and compute maximum queue length at the bottleneck node.

Since we consider a system with N flows, we scale the capacities of the bottleneck node with the input and output capacities of the bottleneck node being Nc_I and Nc_O respectively. This scaling ensures that the *steady-state rate allocated to each user is invariant with the number of sessions.* Physically, this would correspond to a bandwidth scaling at the bottleneck. To mathematically reflect such a scaling, we scale the congestion price appropriately such that the equilibrium rate for each user is invariant with N. In other words, the fraction of flow marked at the bottleneck, $(\frac{1}{Nc_I} + \frac{1}{Nc_O})\lambda^{(N)}(\cdot)$, is invariant with N, where $\lambda^{(N)}(z) = \beta^{(N)}p^{(N)}(z)$ is the congestion price at the bottleneck in the N-th system, see (5.8) (this is analogous to scaling the marking function in [8, 26]). Hence, the dynamics of each flow $x_j(t)$ at the N-th system is given by

$$\dot{x_j}(t) = \kappa\Big[w_j - \beta x_j(t - d_R)\Big(\frac{1}{Nc_I} + \frac{1}{Nc_O}\Big)Np\Big(\sum_{k=1}^{N} x_k(t - d_R)\Big(\frac{1}{Nc_I} + \frac{1}{Nc_O}\Big)\Big)\Big] \tag{5.15}$$

Then, the steady-state rate of flow j, denoted by x_j^* is given by

$$x_j^* \;\; = \;\; \frac{w_j x^*}{\beta(x^*(1/c_I + 1/c_O) - \tilde{t})},$$

where x^* is the average steady-state rate over all flows, and is invariant with N. Note that $\tilde{t}$ is the virtual time constant of the marking function (5.9), and $p(\cdot)$ is the marking function of the bottleneck node.

Further, we assume that the equilibrium rates are stable, (i.e., $\sum_i x_j^\star(\frac{1}{Nc_I} + \frac{1}{Nc_O}) < 1 - \epsilon_b$, where ϵ_b is the MAC efficiency at the bottle-neck node). This ensures that the queue-lengths are zero at the bottleneck link when the system is at equilibrium (however, queues will build up during transients due to network propagation delay).

We finally comment that we have assumed that the feedback (marks) do not experience congestion, and that the delay in the feedback is solely due to propagation delays. As we have per-flow queueing, a packet implementation to approximate this could be the following. When congestion occurs at a node, instead of marking the incoming packet (implemented via setting the ECN (Explicit Congestion Notification) bit [21]), one could mark the head-of-line (outgoing) packet in the queue of the corresponding user. This would ensure that the queueing delays are minimized for the feedback, and that the source gets the appropriate feedback. Such a scheme seems feasible in a wireless context, as per-flow queueing is expected to be implemented for scheduling as well as physical layer reasons [75, 81, 82]. Further, the number of flows in this network is expected to be of a smaller scale than that in the wired network.

Let $x(t) = \frac{1}{N}\sum_{j=1}^{N} x_j(t)$ and $w = \frac{1}{N}\sum_{j=1}^{N} w_j$. By summing (5.15) across all sessions, we obtain

$$\begin{aligned} \dot{x}(t) &= \kappa\Big[w - \beta\Big(x(t-d_R)\Big(\frac{1}{c_I}+\frac{1}{c_O}\Big) - \tilde{t}\Big)^+\Big] \\ &= \tilde{\kappa}[\tilde{w} - (x(t-d_R) - \tilde{c})^+], \end{aligned} \tag{5.16}$$

where $\tilde{c} = \frac{\tilde{t}}{\left(\frac{1}{c_I}+\frac{1}{c_O}\right)}$, $\tilde{\kappa} = \kappa(\frac{1}{c_I}+\frac{1}{c_O})\beta$, and $\tilde{w} = w/(\beta(\frac{1}{c_I}+\frac{1}{c_O}))$.

Next, for each time t, under the end-to-end control scheme, let us denote the average (over flows) queue length (across sessions) at the bottleneck node by $q^e(t)$, and the average input and output rates by $x_I(t)$ and $x_O(t)$ respectively. Observe that congestion occurs if $\frac{x_I(t)}{c_I^b} + \frac{x_O(t)}{c_O^b} > 1$, where $c_I^b = c_I/(1-\epsilon_b)$ and $c_O^b = c_O/(1-\epsilon_b)$. Since the constant scaling of c_I and c_O does not affect the analysis result (Lemma 5.7.1 and Proposition 5.7.1), we assume that $\epsilon_b = 1$ for notational simplicity.

Further, observe that $x_I(t) \le c_I$. We now describe the dynamics of the queue length process. We consider several cases:

(i) $\frac{c_I c_O}{c_I+c_O} < x_I(t) \le c_I$

$$\begin{aligned} \dot{q}^e(t) &= x_I(t) - x_O(t) = c_I\frac{x_I(t)}{c_I} - c_O\Big(1 - \frac{x_I(t)}{c_I}\Big) \\ &= \Big(\frac{c_I+c_O}{c_I}\Big)\Big[x_I(t) - \frac{c_I c_O}{c_I+c_O}\Big] \end{aligned} \tag{5.17}$$

(ii) $x_I(t) \leq \frac{c_I c_O}{c_I + c_O}$ and $q^e(t) > 0$

The dynamics of $\dot{q}^e(t)$ are identical to that in Case (i).

(iii) $x_I(t) \leq \frac{c_I c_O}{c_I + c_O}$ and $q^e(t) = 0$

In this case, as the buffer at the bottleneck node is empty, and there is no congestion, we have $\dot{q}^e(t) = 0$.

Thus, with

$$\dot{\tilde{q}}^e(t) = \frac{\dot{q}^e(t)}{(c_I + c_O)/c_I}, \tag{5.18}$$

we have

$$\dot{\tilde{q}}^e(t) = \begin{cases} x_I(t) - c & \text{if} \quad \tilde{q}^e(t) > 0, \\ (x_I(t) - c)^+ & \text{if} \quad \tilde{q}^e(t) = 0, \end{cases} \tag{5.19}$$

where $c = \frac{c_I c_O}{c_I + c_O}$.

We next derive the "worst-case" peak initial queue length at the bottleneck node under the end-to-end controller as well as a hop-by-hop controller, and with the system starting from rest (i.e., the value of the first local-maximum of the queue length at the bottleneck, when all initial conditions are zero). Let us define ${Q_{max}}^e(H, \eta)$ to be the (unscaled) initial maximum queue length at the bottleneck node in the end-to-end control with the one-hop round-trip delay and the number of hops (links) for each session being η and H, respectively (thus, the end-to-end round-trip delay is ηH). Also let ${q_{max}}^e(H, \eta) = {Q_{max}}^e(H, \eta)/N$ be the peak (initial) queue length for the scaled system defined by (5.16) and (5.19).

As discussed earlier, we assume that the equilibrium rate is stable. Thus, we henceforth let γ be the node utilization at the bottleneck node, i.e., $x^\star = \gamma c$, where $0 < \gamma < 1$. Further, let

$$\bar{q}^e_{max}(1, d) = \left(\tilde{\kappa}\tilde{w} + \left(\frac{\tilde{\kappa}\tilde{w}}{c(1-\gamma)}\right)^2\right)d^2 + \left(c(1-\gamma) + \frac{\tilde{\kappa}\tilde{w}}{c(1-\gamma)}\right)d + \frac{c^2(1-\gamma)^2}{\tilde{\kappa}\tilde{w}} + c(1-\gamma). \tag{5.20}$$

With these definitions, we have

Lemma 5.7.1. *Fix any $d > 0$. Then, we have $q^e_{max}(1,d) \leq \bar{q}^e_{max}(1,d)$, and for any $\alpha \in (0,1)$, $\exists M_o$ with $M_o \geq 1$, such that $\forall M \geq M_o$, $M^\alpha \bar{q}^e_{max}(1,d) \leq {q_{max}}^e(M,d)$.*

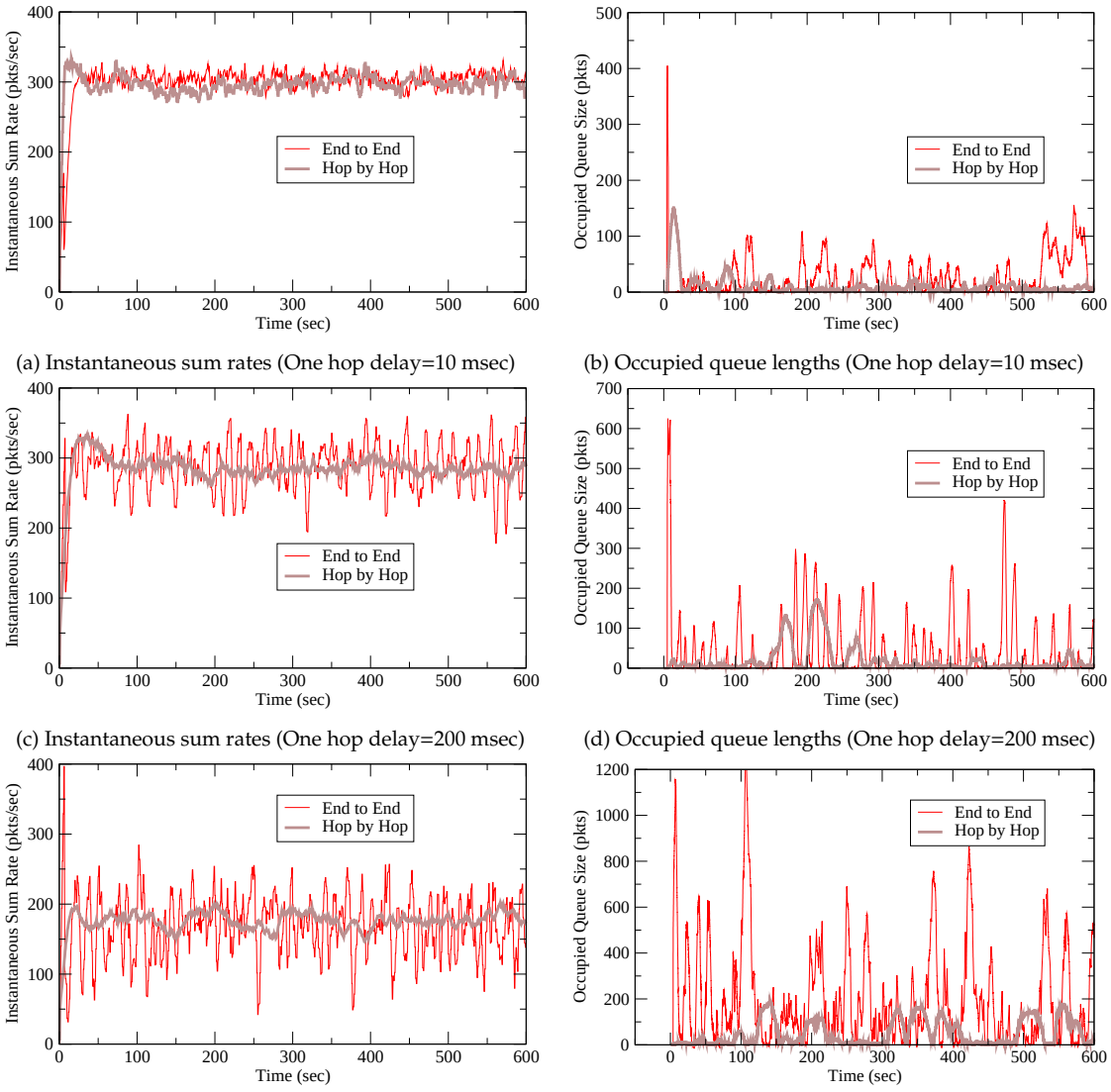
The proof consists of two parts. We first derive an upper bound on the queue length for a 1-hop network (with delay d) by *(i)* observing that the source controller increase rate cannot increase faster than w (i.e., $\dot{x}(t)) \leq w$), and *(ii)* by providing a lower-bound on the rate of decrease of the arrival rate, once the controller rate has peaked. More precisely, for time $t \geq t_5 + d$, where t_5 is the time at which the controller rate peaks (see Figure 5.6), we show $\dot{x}(t) \leq -\epsilon$ for some explicitly constructed $\epsilon > 0$. This upper bound is denoted by $\bar{q}^e_{max}(1,d)$, and formally given by (5.20). We next derive a lower bound on ${q_{max}}^e(M,d)$, by appropriately bounding the end-controller transmission rates. Using this lower-bound, and the upper bound for the 1-hop system, the lemma is proved. The details of the proof is presented in the Appendix.

Using this result, we prove the main result of this section. Similar to ${Q_{max}}^e(H,\eta)$ and ${q_{max}}^e(H,\eta)$, let us define ${Q_{max}}^h(H,\eta)$ to be the unscaled maximum queue length (due to initial transients) with the *hop-by-hop control*, and ${q_{max}}^h(H,\eta)$ to be the corresponding scaled queue-length. We then have

Proposition 5.7.1. *Fix any $d > 0$. Then, for any $\alpha \in (0,1)$, $\exists M_o$ with $M_o \geq 1$, such that $\forall M \geq M_o$, ${q_{max}}^h(M,d) \leq {q_{max}}^e(M,d)/M^\alpha$.*

Proof. The proof proceeds analogously as that in Lemma 5.7.1. As the control loop for the hop-by-hop controller has round trip delay d, the upper-bound derived for the 1-hop end-to-end controller with round-trip delay of d can be shown to also hold here, i.e., ${q_{max}}^h(M,d) \leq \bar{q}^e_{max}(1,d)$. Now, from Lemma 5.7.1, the desired result follows. □

Remark 5.7.1. *The tree network in Figure 5.4-(b) is a special case of the feed-forward network, such that each session has a separate path. The tree architecture could be used in a community roof-top wireless network, where a single wired based station could be used to provide community Internet access. Tree architectures are also popular in Bluetooth based ad hoc networks [92, 93], as well as in broadcasting applications [94]. In Section 5.8, we will see that we achieve significant gains for the tree network even with only five flows.*



(a) Instantaneous sum rates (One hop delay=10 msec)

(b) Occupied queue lengths (One hop delay=10 msec)

(c) Instantaneous sum rates (One hop delay=200 msec)

(d) Occupied queue lengths (One hop delay=200 msec)

(e) Instantaneous sum rates (One hop delay=200 msec, with 10% background ON-OFF flows)

(f) Occupied queue lengths (One hop delay=200 msec, with 10% background ON-OFF flows)

Figure 5.5: Instantaneous sum rates and occupied queue lengths (at the bottleneck node) for both end-to-end and hop-by-hop controllers

5.8 Simulation Results

In this section, we present simulation results that compare the hop-by-hop algorithm with the end-to-end algorithm, and show that there is a significant decrease in the peak load with the hop-by-hop algorithm. We consider packet level simulations using the ns-2 [33] simulator. We have made suitable modifications in ns-2 to support hop-by-hop as well as

end-to-end controllers with a MAC.

5.8.1 Simulation Setup

The network topology used in the simulation is a tree network shown in Figure 5.4-(b), with five sessions and five hops from the sources and the destinations. Each of the input and output links at the bottleneck node has a bandwidth of 5 Mbps. Since we use a fixed size packet of 1000 bytes, this corresponds to 625 pkts/sec. Thus, the equilibrium rate over one session at the bottleneck is 62.5 pkts/sec.

In our ns-2 implementation, we have assumed a separate side channel for communicating prices (with a bit rate of 1 bit per data packet) that enables a simple implementation. In practice, this price information can be embedded in ACK packets. We used a fixed data packet size of 1000 bytes. For every one data packet, the associated ACK packet contains one "ECN" bit, which represents marked/unmarked status. To compute the price information at every node, each node measures outgoing/incoming data rate over all incident links on the corresponding node in the following manner. For outgoing rates, the node knows the transmission rates (from the congestion controller at the node); and for incoming rates, the node measures rates by computing inter-packet times (using a sliding window mechanism).

The following parameters have been chosen for the congestion controllers, and marking functions at the intermediate routers, such that the node utilization ratio is about 95%: $\kappa = 0.5,\ w = 21,\ \tilde{t} = 0.25,\ \beta = 30$. From (15), we have

$$x^\star\Big(\frac{1}{c_I} + \frac{1}{c_O}\Big) = \tilde{t} + \frac{w}{\beta} = 0.25 + \frac{21}{30} = 0.95. \tag{5.21}$$

Thus, we have the 95% node utilization.

In our implementation, we approximate the fluid model for a MAC with a time-division MAC. The MAC protocol implemented in the simulation is a simple centralized TDMA MAC protocol (a centralized scheduler that distributes the requested number of time-slots to each link). A time slot interval is chosen to be the transmission time of a single packet. If the incoming/outgoing traffic imposed on a node violates the timing constraint,

as discussed in Section V in the paper, we give higher priority to the incoming traffic to the nodes (in this case queueing occurs on the output link buffer).

5.8.2 Simulation Results

Figures 5.5(a) and (c) plot the instantaneous sum rate of five sessions for 10 msec and 200 msec of one-hop delay, respectively. The 10 msec case corresponds to an efficient MAC, where the hop-delay primarily occurs due to the propagation delay and physical layer air interface. On the other-hand, the case where the hop delay is 200 msec corresponds to a network with large per-hop delays (possibly due to the MAC implementation). In both of these cases (Figures 5.5(a) and (c)), we see that the transmission rates (with the end-to-end and hop-by-hop controllers) oscillate about the corresponding fixed point. As expected, we observe that for the large delay case, the end-to-end algorithm leads to larger oscillations than the hop-by-hop algorithm.

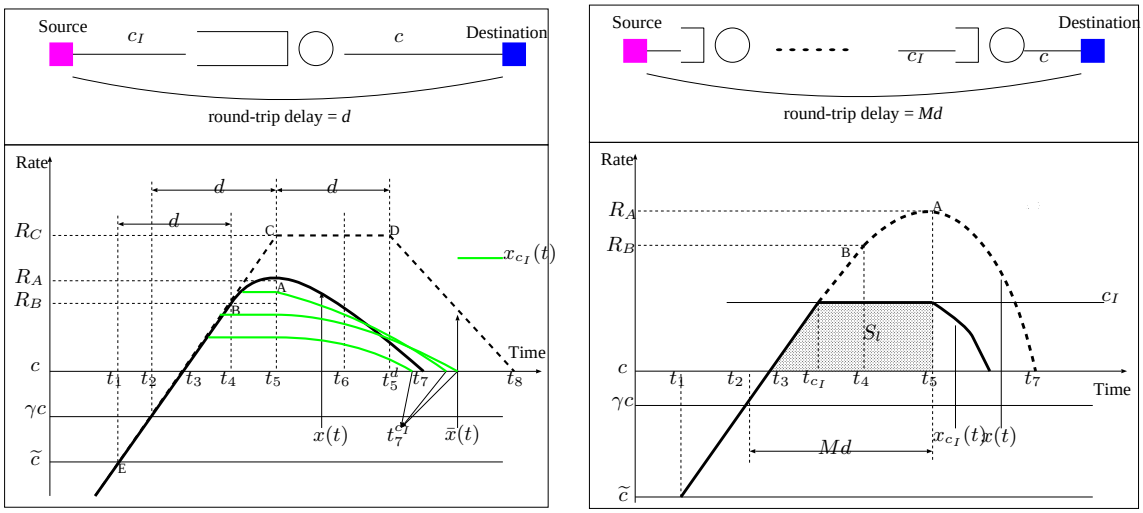
In Figures 5.5(b) and (d), we plot the evolution of the bottleneck queue length (measured every 20 msec). The spatial spreading effect is clearly illustrated by these plots, where we see that the peak buffer size with the hop-by-hop controller is much smaller than that of the end-to-end controller. These fluid and packet simulations indicate that significant gains are to be had with the hop-by-hop scheme, and validate our analytical results.

We also provide a plot with short background flows (modeled by ON-OFF processes) in Figures 5.5(e) and (f). The only difference from the previous setup is that we have background ON-OFF flows, whose mean rate is set to be 10% of the equilibrium rates of controlled flows, and their burst and idle times are set to be 50 msec each. Given a w, we suitably chosen β and $\tilde{t}$ such that the node utilization is about 95%. The values are chosen in a similar manner to (5.21). Figures 5.5(e) and (f) show that again, the hop-by-hop scheme outperforms an end-to-end scheme.

Appendix: Proof of Lemma 5.7.1

Proof. In this proof, we will show the following: For any fixed $d > 0$,

(i) ${q_{max}}^e(1, d) \leq \tilde{q}^e_{max}(1, d)$



(a) Upper bound on peak queue length $q_{max}{}^e(1,d)$ (b) Lower bound on peak queue length $q_{max}{}^e(M,d)$

Figure 5.6: Upper bound and lower bound on peak queue length

(ii) $\exists M_o$ such that $\forall M \geq M_o$, $q_{max}{}^e(M,d) \geq KMd$, where K is a positive finite constant.

Suppose that **(i)** and **(ii)** are true. Then, from the definition of $\bar{q}^e_{max}(1,d)$ (see (5.20)), for any $0 < \alpha < 1$, and sufficiently large M, we have

$$\begin{aligned} M^\alpha q_{max}{}^e(1,d) &\leq M^\alpha \bar{q}^e_{max}(1,d) \\ &\leq KMd \leq q_{max}{}^e(M,d). \end{aligned}$$

Thus, it suffices to show **(i)** and **(ii)** to complete the proof.

Proof of (i): Consider a network of a single link with the end-to-end round-trip delay of each session being d. As discussed earlier, we have the equilibrium rate $x^\star = \gamma c$, where γ is the node utilization with $0 < \gamma < 1$. Thus, at the steady state we have

$$\tilde{w} = \gamma c p(\gamma c) = \gamma c - \tilde{c}. \tag{5.22}$$

Recall that $p(\cdot)$ is the marking function of the bottleneck node (see (5.15)).

Let us define the following time epochs (see Figure 5.6).

$$t_1 = \inf\{t > 0 : x(t) > \tilde{c}\},\ t_2 = \inf\{t > t_1 : x(t) > \gamma c\},$$

$$t_3 = \inf\{t > t_2 : x(t) > c\},\ t_4 = t_1 + d,\ t_5 = t_2 + d,$$
$$t_6 = t_4 + d,\ t_7 = \inf\{t > t_5 : x(t) < c\} \tag{5.23}$$

We first consider an end-to-end system with an input rate constraint c_I (which could be arbitrarily large), as shown in Figure 5.6-(a). Due to the input constraint c_I, the actual rate arriving at the bottle-neck node, which we denote by $x_{c_I}(t)$, could be different from $x(t)$, depending on c_I and R_A (see Figure 5.6-(a)). We will upper-bound $x(t)$ (and $x_{c_I}(t)$) by a trajectory $\bar{x}(t)$, and use this bound to compute an upper bound on the peak queue length. We consider the following two cases: (a) $c_I \geq R_A$ and (b) $c_I < R_A$.

Case (a): $c_I \geq R_A$

In this case, $x(t)$ is the actual input arrival rate at the bottle-neck node. Now, observe that the peak queue length at the bottleneck node is given by

$$q_{max}{}^e(1, d) \quad = \quad \int_{t_3}^{t_7} (x(t) - c)\, dt$$

We assume that the initial condition satisfies $x(s) \leq \tilde{c}$, $\forall s \leq 0$.

We can see that $x(t)$ achieves the maximum (i.e., R_A) at t_5, since $\dot{x}(t_5) = 0$. This follows from the fact that

$$\dot{x}(t_5) \quad = \quad \tilde{\kappa}(\tilde{w} - x(t_2)p(x(t_2))) = \tilde{\kappa}(\tilde{w} - \gamma c p(\gamma c)) = 0,$$

and from (5.22). For $t \in [t_1, t_4]$, let $\bar{x}(t) = x(t)$. Then, by the assumption on the initial condition

$$\dot{x}(t) = \dot{\bar{x}}(t) = \tilde{\kappa}\tilde{w}, \quad t \in [t_1, t_4] \tag{5.24}$$

Next, let $\dot{\bar{x}}(t) = \tilde{\kappa}\tilde{w},\ t \in (t_4, t_5]$ with $\bar{x}(t_4) = R_B$ (i.e., straight-line extension of $\bar{x}(t), t \in [t_1, t_4]$ until t_5). Then, we have

$$x(t) < \bar{x}(t), \quad t \in (t_4, t_5], \tag{5.25}$$

since for $t \in (t_4, t_5]$

$$\begin{aligned} \dot{x}(t) &= \tilde{\kappa}(\tilde{w} - x(t-d)p(x(t-d)) \\ &< \tilde{\kappa}\tilde{w} = \dot{\bar{x}}(t) \end{aligned}$$

Let $t_5^d = t_5 + d$. Further, for $t \in (t_5, t_5^d]$ let $\bar{x}(t) = \bar{x}(t_5)$. Then, we have

$$x(t) \leq R_A < \bar{x}(t_5) = \bar{x}(t), \quad t \in (t_5, t_5^d], \tag{5.26}$$

Note that we have the following two cases: (i) $t_5^d \leq t_7$ and (ii) $t_5^d > t_7$. First, consider the case of $t_5^d \leq t_7$ (Figure 5.6-(a)). For $t \in (t_5^d, t_7]$, let us define $\bar{x}(t)$ with $\dot{\bar{x}}(t) = \tilde{\kappa}(\tilde{w} - (c - \tilde{c}))$.

Then, using the fact that $x(t-d) > c,\ t \in (t_5^d, t_7]$, we have $\dot{\bar{x}}(t) > \dot{x}(t) = \tilde{\kappa}(\tilde{w} - (x(t-d) - \tilde{c}))$, leading to the fact that

$$x(t) < \bar{x}(t), \quad t \in (t_5^d, t_7]. \tag{5.27}$$

Second, if $t_5^d > t_7$, this case corresponds to (5.26).

Case (b): $c_I < R_A$

In this case, $x_{c_I}(t)$ is the actual input arrival rate at the bottle-neck node. Let $t_7^{c_I} = \inf\{t > t_5 : x_{c_I}(t) < c\}$. Note, however, that $\forall t \in [t_3, t_7^{c_I}]$, we have $x_{c_I}(t) \geq c$. Thus, the bound (i.e., $\bar{x}(t)$) constructed in Case (i) based on $x(t)$ also holds for $x_{c_I}(t)$, i.e.,

$$x_{c_I}(t) \leq \bar{x}(t), \quad t \in [t_3, t_7^{c_I}] \tag{5.28}$$

From (5.25), (5.26), (5.27), and (5.28), $\bar{x}(t)$ is an upper bound on $x(t)$ or $x_{c_I}(t)$ irrespective of the position of c_I.

By the straight line extension of $\bar{x}(t)$ until it crosses c, we define $t_8 \triangleq \inf\{t > t_5^d : \bar{x}(t) < c\}$.

Note that we have

$$\begin{aligned} t_5 - t_3 &= d - (t_3 - t_2) = d - \frac{c(1-\gamma)}{\tilde{\kappa}\tilde{w}} \\ R_C - c &= \tilde{\kappa}\tilde{w}(t_5 - t_3) \\ t_8 - t_5^d &= \frac{R_C - c}{\tilde{\kappa}(c - \tilde{c} - \tilde{w})} = \frac{R_C - c}{c(1-\gamma)}. \end{aligned}$$

Thus, the upper bound on the peak occupied queue length for a fixed d is given by:

$$\begin{aligned} {q_{max}}^e(1,d) &\leq \int_{t_3}^{t_8} (\bar{x}(t) - c)\, dt \\ &= \frac{1}{2}(t_5 - t_3 + t_5^d - t_5 + t_8 - t_5^d)(R_C - c) \\ &\leq \left(\tilde{\kappa}\tilde{w} + \left(\frac{\tilde{\kappa}\tilde{w}}{c(1-\gamma)}\right)^2\right)d^2 + \left(c(1-\gamma) + \frac{\tilde{\kappa}\tilde{w}}{c(1-\gamma)}\right)d \\ &\quad + \frac{c^2(1-\gamma)^2}{\tilde{\kappa}\tilde{w}} + c(1-\gamma) \\ &= \tilde{q}_{max}^e(1,d). \end{aligned} \tag{5.29}$$

This completes the proof of **(i).**

Proof of (ii): Next, we show **(ii)**, which describes ${q_{max}}^e(M,d)$ for *sufficiently large M*. In this setup, we consider an end-to-end controlled system with its round-trip delay being Md. Thus, the time epochs described in (5.23) hold for the round-trip delay Md. Figure 5.6-(b) explains this setup.

Recall that $R_B = x(t_4)$. As $R_B - \tilde{c} = (t_4 - t_1)\tilde{\kappa}\tilde{w}$, we have

$$R_B = Md\tilde{\kappa}\tilde{w} + \tilde{c} \tag{5.30}$$

Note that we derive ${q_{max}}^e(M,d)$ for *sufficiently large M*. Then, it is clear that for a sufficiently large M and any finite c_I, we have $c_I \leq R_B$. Thus, $x_{c_I}(t)$ is the actual arrival rate to the bottle-neck node with its input constraint of c_I. Instead of computing the exact peak queue length, we lower-bound it by computing the area of the shaded region (denoted by S_l) in Figure 5.6-(b).

Let t_{c_I} be the first time after t_3 such that $\bar{x}(t)$ hits c_I. By definition of time epochs, $t_5 - t_3 = Md$ (see (5.23)), and we have

$$t_{c_I} - t_3 = \frac{c_I - c}{\tilde{\kappa}\tilde{w}}$$

Then, we have

$$S_l = \frac{1}{2}\Big(2(t_5 - t_3) - (t_{c_I} - t_3)\Big)(c_I - c)$$

$$
\begin{aligned}
&= \frac{1}{2}\left(2Md - \frac{c_I - c}{\tilde{\kappa}\tilde{w}}\right)(c_I - c) \\
&= Md(c_I - c) - \frac{1}{2}\frac{(c_I - c)^2}{\tilde{\kappa}\tilde{w}} \qquad (5.31)
\end{aligned}
$$

Thus, we can find a positive constant K, such that $\forall M \geq M_o$

$$
{q_{max}}^e(M, d) \quad \geq \quad KMd. \qquad (5.32)
$$

Then, the proof of **(ii)** follows from (5.32). □

Chapter 6

Randomized Contention-Aware MAC Scheduling

6.1 Overview

Aggregate traffic loads and topology in multi-hop wireless networks may vary slowly, permitting MAC protocols to 'learn' how to spatially coordinate and adapt contention patterns. Such an approach could reduce contention, leading to better throughput and energy consumption. To that end we propose a new family of distributed TDMA MAC scheduling algorithms combining synchronous two-level priority RTS/CTS handshaking with randomized time slot selection. We prove that for any fixed admissible load such algorithms converge to a feasible schedule exponentially fast, and so are *throughput-optimal.* Furthermore, by adaptively biasing time-slot selection probabilities based on past history, one can develop variations that are also provably throughput-optimal and exhibit better convergence rates. Additionally under moderate loads local changes in load would lead to only local changes in contention patterns leading once again to fast convergence. This makes the case for adopting such protocols in wireless multi-hop networks, where aggregate loads and network topology are slowly varying.

6.2 Introduction

The design of MAC protocols for wireless multi-hop networks has received much attention over the last decade. These protocols can be broadly classified into contention-based schemes and scheduling-based schemes (see [95] for a survey).

Contention-based schemes (e.g., IEEE 802.11 [76]) are based on random channel access, and enable nodes to transmit data asynchronously, with retransmission (after suitable time-out) if the transmission is unsuccessful. The major limitation of contention-based schemes is that the throughput significantly degrades with increasing load due to collisions. On the other hand, scheduling-based schemes allocate channel resources (using either centralized or distributed strategies) in order to minimize resource contention. While scheduling-based schemes have the advantage that there is no loss in throughput due to collisions, contention-based schemes have been popular due to the simplicity of implementation. In particular, scheduling-based schemes seem to be inflexible and not scalable to load/topology changes, mainly because such load/topology changes force the existing scheduling decision to be disseminated to the entire network (see the related work later).

In this chapter, we study a MAC scheduling algorithm, which leverages the advantages of both schemes. We restrict our discussion to the case where shared resources are time-slots (i.e., a TDMA system). However, it is well-known in literature that resource allocation algorithms for a TDMA system immediately extend to FDMA or CDMA systems as long as the resource satisfies an orthogonality property (i.e., "non-overlapping" resources such as different time-slots, or orthogonal codes).

To that end, we propose a *synchronous* contention-based MAC scheduling algorithm, which self-adapts to changes in traffic load and network topology, converging, if possible, to a conflict-free schedule by exchanging synchronous control messages, as data transmissions occur simultaneously (see Figure 6.1). Thus, our algorithm has reasonably high throughput even during transients, and automatically adapts to load changes without any explicit notification, by 'learning' *local* contention patterns. Further, our algorithm is a distributed one, where only *one-hop* control message exchange is required.

Our research is motivated by the following factors:

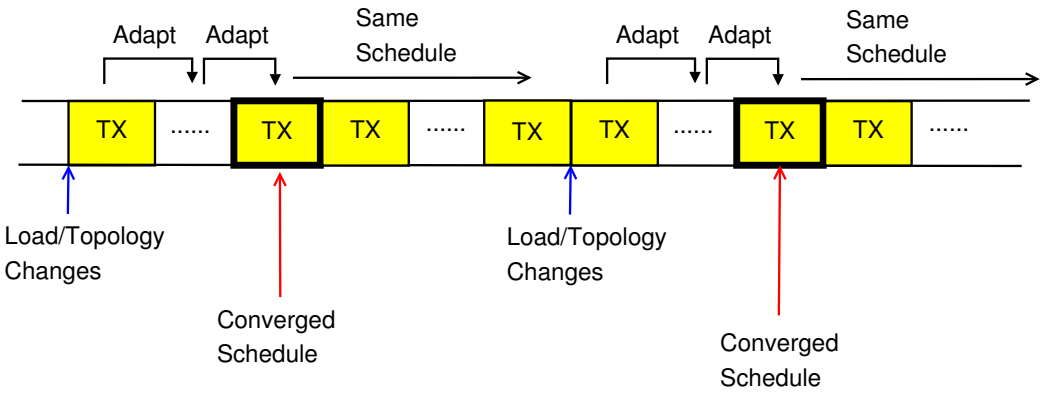


Figure 6.1: Load/Topology-Adaptive TDMA MAC Scheduling

(i) Slowly varying loads/topologies: Our premise in this chapter is that although individual traffic loads may change quickly, the aggregates on congested nodes may, in many relevant applications, change more slowly. Similarly, node mobility leads to changes in topology (and thus changes in load), but again these changes might be slow enough to permit a MAC protocol to learn and exploit the offered traffic characteristics so as to quickly realize conflict-free schedules.

(ii) Learning contention patterns: A TDMA MAC protocol is known to achieve a high throughput after it finds a conflict-free schedule. However, whenever load or network topology changes, it has to re-initiate a "scheduling-decision" phase to find a conflict-free schedule. On the other hand, a contention-based protocol (e.g., IEEE 802.11) asynchronously transmits data, enabling easier implementation and better robustness to load or network topology changes. However, its throughput significantly degrades with increasing loads. Our goal in this chapter is to leverage the advantages of both protocols, to realize high throughput and robust adaptability to load and/or network topology changes.

By using *synchronous* contention, we expect to learn contention patterns, such that time-slot allocation (chosen locally by nodes) can become close to an "efficient" schedule by progressively learning the past contention patterns. Synchronous contention [96–98] is known to be a good strategy for efficient channel utilization, since it protects data transmissions as well as acknowledgments, leading to eliminating the need for maintaining protocol states (e.g., NAVs in IEEE 802.11 [76]), as compared to asynchronous schedules.

Further, it provides a better framework to support priority access and better QoS [99]. In part, synchronous contention is crucial for throughput-optimality and enables us to break "deadlock" using a multi-level priority control messages (see Section 6.4). The trade-off is the additional effort to synchronize nodes with equal time. However, this condition of tight synchronization can be relaxed with minor performance decrease [99].

(iii) Guaranteeing high throughput and fast convergence: If an adaptive MAC is to be useful, then high throughput and fast convergence (to a conflict-free schedule) should be guaranteed. Otherwise, most of the time will be devoted to searching for a conflict-free schedule with possibly low throughput. In particular, fast convergence is indispensable for tracking the time-varying loads and topologies. However, such algorithms to date [100–103] do not *provably* guarantee high throughput and fast convergence, or assume limited network topology (e.g., tree) as well as the restricted collision model. Thus, the challenge remains to devise an algorithm, which provably and quickly converges a conflict-free schedule for any feasible load, irrespective of network topology (i.e., throughput-optimal[1]).

In a typical TDMA system, time is divided into transmissions slots (time-slots), which are grouped into frames. We consider a TDMA system, where each link is subject to an offered traffic load, which is typically represented by the number of time-slots over a frame. Depending on the service supported by the network, information on the offered load could either be explicitly given to the nodes or be measured by the nodes. If we have a guaranteed-service network based on a resource reservation signaling (e.g., RSVP [104]), the amount of load could be known a priori by nodes in the path of a reserved flow. However, in a typical best-effort service network, the amount of load is not explicitly provided to the nodes, but the nodes could know the offered load by measuring/estimating it over a suitable time-period. Because the loads might exhibit some variation, or measurement might be noisy, a node might use an upper estimate for it (i.e., overbook) such as $\hat{\rho}_l + \hat{\delta}_l$, where $\hat{\rho}_l$ and $\hat{\delta}_l$ are the estimated mean and standard deviation for the offered load on link l.

The problem of finding a conflict-free schedule in a TDMA based wireless multi-

[1]A link scheduling scheme is said to be *throughput-optimal* if it can find a conflict-free schedule for any feasible offered load. An offered load is *feasible* if there exists a conflict-free schedule (see Section 6.3.2 for formal definitions).

hop network, has been an active research topic. Prior work can be classified into two categories: *(i)* link scheduling [89,105–109] and *(ii)* node scheduling [110–116]. These have been studied mainly based on considering the associated edge or node coloring problems. The underlying assumption for these studies is that every link has the same traffic demand (i.e., uniform traffic load or infinitely backlogged data). For a non-uniform load, the link scheduling problem can be formulated by two-hop edge coloring in a *multi-graph*[2], where centralized sub-optimal solutions (in terms of time-complexity) are available [117, 118]. Even though two-hop edge-coloring based approach has been popularly used for TDMA network, it is known to be an inappropriate problem transformation for *link* scheduling algorithm, since two simultaneous transmissions in the two-hop distance away do not always conflict with each other (i.e., exposed node problem [119]) (see [106] for details). In this chapter, we consider a *link scheduling* algorithm for a *non-uniform* load in the network.

We note that all the above-mentioned strategies are for *static* scenarios, where the scheduling-decision phase and data transmission phase are separated. Thus, any network topology or load changes lead to a new scheduling-decision phase. Further, even when these are implemented as distributed algorithms, every node should be notified of the event of a change by a broadcasting of control messages. Most of research on this area assumes that such control messages are successfully transferred to nodes contention-free, which seems to be unrealistic in the resource-constrained wireless multi-hop networks. Our work differs from the above-mentioned work in that our algorithm adapts to load or topology changes without explicit notification of such changes.

An alternate approach is to devise a dynamic scheduling algorithm, where data transmission and scheduling-decision (time-slot allocation) occur simultaneously. Several dynamic algorithms have been proposed [100–103]. However, they require either two-hop connectivity information [100], or are not provably throughput-optimal [100–102]. The work in [103] is limited to networks with only "node-exclusive conflict model" (i.e., primary conflict), and is only shown to provably converge for a tree network topology. Our work also differs from these in that for *any* arbitrary network and feasible load, our algorithm converges to a conflict-free scheduling allocation (throughput-optimal), as proved in

[2]A graph whose edges are unordered pairs of vertices, and the same pair of vertices can be connected by multiple edges.

Section 6.5.

In particular, our work is closest to FPRP (Five-Phase Reservation Protocol) [100], which jointly and simultaneously performs the tasks of channel access and scheduling. However, our work differs from FPRP in that (i) FPRP considers only node scheduling and uses two-hop control messages to find a good schedule, (ii) more importantly, FPRP is not throughput-optimal, i.e., even if there is an appropriate schedule satisfying the offered load in the network, FPRP sometimes get dead-locked in a bad schedule.

Another interesting approach is topology-independent scheduling [120, 121], where the basic idea is to generate a schedule, such that each node has at least one transmission slot conflict-free (for any topology change) by overbooking time-slots in a larger frame than optimally required, leading to a decrease of channel utilization.

6.2.1 Main Contributions and Organization

The main contributions of this chapter are as follows:

(i) We propose a synchronous contention-based load/topology-adaptive TDMA link scheduling algorithm (DCAMA: Dynamic Contention-Aware Multiple Access). In the DCAMA algorithm, two-level RTS/CTS synchronous control message are used together with randomized time-slot selection at each links. We prove that the DCAMA always converges to a conflict-free schedule (if there exists one), and its rate of convergence is exponentially fast, for any feasible load and any arbitrary network topology, and so is throughput-optimal.

(ii) The importance of DCAMA algorithm is that it could act as a base-line algorithm, whose variations are also provably throughput-optimal and their rate of convergence is also exponential. Thus, we propose an adaptive variation to the DCAMA algorithm (ADCAMA: Adaptive DCAMA), which adaptively biases slot selection probabilities based on contention histories of the previous m frames. We prove that the ADCAMA algorithm also converges to a conflict-free schedule exponentially fast, and by simulation we show that only a three-frame history is necessary to significantly improve the rate of convergence and the transient throughput, resulting in a good adaptation

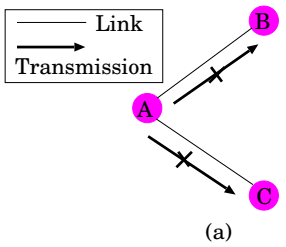


Figure 6.2: (a) Unicasting and (b) Half-Duplex.

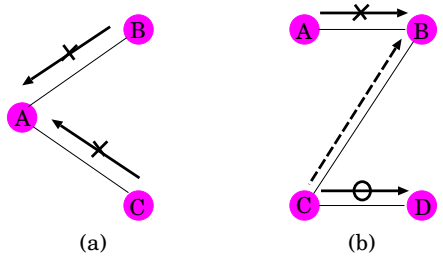


Figure 6.3: (a) Primary Conflict and (b) Secondary Conflict.

to load/topology changes.

The paper is organized as follows. We begin with a description of the system model in Section 6.3. Next, in Section 6.4, we describe the DCAMA algorithm and its properties. In Section 6.5, we prove that the DCAMA converges to a conflict-free schedule for any feasible load, irrespective of network topology. We then propose an adaptive heuristic (adaptive DCAMA) to improve the convergence rate (Section 6.6). Finally, in Section 6.7, we validate the results using simulations.

6.3 System Model and Problem Formulation

6.3.1 System Model

We model the wireless multi-hop network by a graph $\mathcal{G}(\mathcal{L},\mathcal{V})$, where $\mathcal{L} = \{1,\cdots,|\mathcal{L}|\}$ denotes a set of directional links, and $\mathcal{V} = \{1,\cdots,|\mathcal{V}|\}$ denotes a set of nodes. We assume that for any link between two nodes there is a counter-part in the opposite direction. The wireless system has a *single* frequency/code, which is available for both data and control message transmission, and there is no separate physical channel for control messages (i.e., in-band signaling). Each node in the system is equipped with an omni-directional antenna, and is synchronized. We assume that each transmission is intended for only one receiver (unicasting constraint), and each node has only a single transceiver (half-duplex radio) (see Figure 6.2).

In our network model, if node $i \in \mathcal{V}$ is within the transmission range of $j \in \mathcal{V}$, then the link from i to j is established (denoted by $i \rightarrow j$). Thus, we have two transmission conflict

scenarios: (i) primary conflicts, where either multiple nodes transmit simultaneously to the same receiver (Figure 6.3(a)), and (ii) secondary conflict, where a node receiving transmission is also within the transmission range of other transmissions not intended for it (Figure 6.3(b)). Further, due to a single transceiver, packet reception and transmission are not allowed to happen simultaneously (Figure 6.2(b)). Finally, the transmission is intended only for one receiver (Figure 6.2(a)). The access problem arises due to the above-mentioned four resource constraints between links[3].

In a TDMA based wireless ad-hoc network, time is divided into transmission slots (*time-slots*), which are grouped into *frames*. A time-slot duration is suitably chosen to accommodate the transmission of one fixed-size packet and includes a guard time corresponding to the maximum differential propagation delay between pairs of nodes in the network. We assume that the frame size in the network is fixed throughout system operation, where the frame size is chosen (heuristically) depending on the number of nodes, network load, and quality-of-service constraints.

We further assume that a node can distinguish between the absence of any transmission and packet collisions (e.g., carrier sensing). For example, in Figure 6.3(a), when B and C are transmitting messages to A simultaneously in a same time-slot, A is unable to decode the message due to collision, but A is able to know that there was transmissions sent to itself.

We do not consider routing and transport-layer end-to-end flows in this study. We focus on "next neighbor transmissions" since multiple access problems depends solely on the next neighbor transmission requirements.

6.3.2 Problem Formulation

With this setup, we denote the *offered-load* on the network by $\vec{\rho} = (\rho_l : l = 1, \cdots, |\mathcal{L}|)$, where ρ_l is the number of the requested time-slots over the link l in a frame, i.e., $\vec{\rho} \in \mathcal{Z}_+^{|\mathcal{L}|}$, where $\mathcal{Z}_+$ is the set of non-negative integers.

The scheduling decision at each frame is represented by a *contention matrix* (CM),

[3]In a typical distributed link scheduling algorithm, a node is responsible for determining slot-schedules for its outgoing links. Thus, the unicasting constraint in Figure 6.2(a) is automatically resolved.

$C(F, \vec{\rho}) = (c_{ls} : l = 1, \cdots, |\mathcal{L}|, s = 1, \cdots, F)$ for a frame-size F and an offered load $\vec{\rho}$, where $c_{ls} = 1$ implies that a transmission is scheduled to contend over the link l on time slot s. For all $l \in \mathcal{L}$, $\rho_l = \sum_{s=1}^{F} c_{ls}$, i.e., the number of contending time-slots is equal to the load offered on that link.

Further, we use $\vec{c}_{l\cdot} = (c_{ls} : s = 1, \cdots, F)$ and $\vec{c}_{\cdot s} = (c_{ls} : l = 1, \cdots, |\mathcal{L}|)$ to refer to the l-th row and s-th column vector of C, respectively. We call $\vec{c}_{l\cdot}$ and $\vec{c}_{\cdot s}$ a *slot schedule* over l and a *link schedule* on time-slot s, respectively. The link l is said to be *satisfied* by $\vec{c}_{l\cdot}$, if all its scheduled transmissions by $\vec{c}_{l\cdot}$ are successful.

Definition 6.3.1. *A contention matrix $C(F, \vec{\rho})$ is said to be feasible if all its links are satisfied. An offered load $\vec{\rho}$ is said to be feasible over a frame size F if there exists a feasible $C(\vec{\rho}, F)$.*

Our primal goal is to devise a *distributed* algorithm, which converges to a feasible schedule (i.e., after it reaches a feasible schedule, it stays at that schedule over successive frames, before any change in traffic loads or network topology) for *any* feasible offered load and *any* network topology (i.e., provably throughput-optimal) under the following constraints: (i) only one-hop control message is permitted between the transmitter and the receiver at a link, and there does not exist a separate contention-free control channel (ii) data transmission and the process to find a feasible schedule are not separated, and (iii) it has to converge a feasible schedule reasonably "fast" such that it follows (asynchronous) load/topology changes well.

6.4 DCAMA Algorithm

6.4.1 Overview

The frame and time-slot structure of the DCAMA algorithm are shown in Figure 6.4. A time-slot is divided into two parts: time to exchange control messages and time to transmit data to the receiver. We describe the DCAMA algorithm by dividing its behavior into two different time-scales: (i) per-frame operation (Section 6.4.2), where at start of each frame, a node determines the slot-schedules for the transmissions over its adjacent outgoing links, following the offered loads, and (ii) per-slot operation (Section 6.4.2), where a node

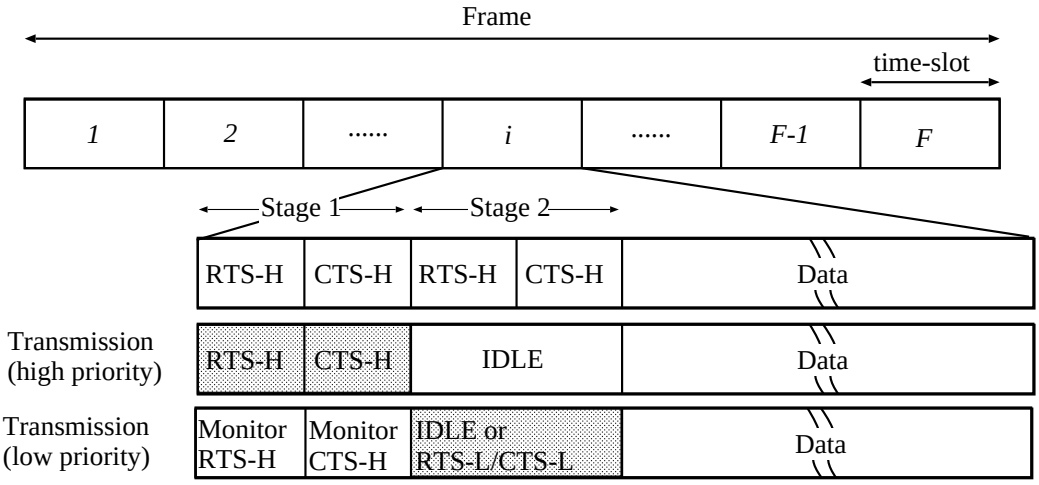


Figure 6.4: Frame and Slot Structure

initiates control message signaling to resolve contentions and transmit data if this time-slot is scheduled by slot-schedules over one of its outgoing link.

To resolve contention, we use a *synchronous* RTS/CTS based mechanism. However, unlike conventional contention based algorithms our contention resolution mechanism has *two-level priority: high and low,* i.e., every scheduled transmission on a slot is assigned one of low or high priority[4]. In other words, control message exchange is decomposed into two stages, where the first and the second stages are used for high and low priority transmissions, respectively. In particular, the transmitters and the receivers of low priority transmissions monitor control message signaling *at the first stage*, and determines whether it has to defer (i.e., release this time-slot) or contend on this time-slot (see Section 6.4.2 for details). An issue with two-level RTS/CTS signaling is that of additional control messages overhead (as compared to the conventional RTS/CTS signaling in the absence of priority). However, as well shall see later in Section 6.7.2, our algorithm does not generate significant additional overheads. Further, due to synchronous contention, we do not require information fields for maintaining states needed by asynchronous protocols (e.g., NAV and DIFS in IEEE 802.11 [76]). In Section 6.7.2, we will quantitatively compute the additional overheads due to two-level RTS/CTS signaling, and show that the performance increase (about 25%)

[4]Throughout this chapter, for notational simplicity, we use RTS-H/CTS-H and RTS-L/CTS-L to refer to control messages with high and low priority level, as needed.

is much higher than the additional signaling overhead (about 3%).

The key mechanisms to achieve a goal to converge to a feasible schedule (for any feasible load and for any network topology) are summarized as follows:

Two level RTS/CTS priority. With a contention mechanism without priority, even for a feasible load, the algorithm could reach a deadlock, and thus it can be trapped in a "bad" schedule, forever (see Figure 6.6 for an example). The two-level priority RTS/CTS mechanism ensures that such a deadlock does not arise.

Synchronized contention. Synchronous contention enables receivers to infer the presence of RTS/CTS transmissions merely by sensing signaling activity over the appropriate time-intervals in a frame (corresponding to the RTS/CTS transmission "slots" within a frame). Note that this does not imply that these messages are successfully decoded by the receiver. Synchronization is useful in conjunction with the two-level priority signaling scheme, as it enables low priority transmissions to release a slot even if a signaling message collision occurs.

Randomized slot selection. The algorithm is randomized in determining slot-schedules (at the next frame) for unsuccessful transmissions. Incorporation of prioritized RTS/CTS mechanism with randomized slot selection strategy enables the system to reach any schedule, and thus to ultimately converge to a feasible schedule.

We additionally introduce a control message priority matrix, $R = (r_{ls} : l \in \mathcal{L},\ s \in F)$ to represent the control message priority, where $r_{ls} = 1$ ($r_{ls} = 0$) if a transmission is scheduled over link l on time-slot s (i.e., $c_{ls} = 1$) and its priority is high (low), and NULL if $c_{ls} = 0$.

6.4.2 Algorithm Description

Determining Slot-Schedules

When each frame starts, a node (say, $v \in \mathcal{V}$) determines the slot-schedules and their RTS/CTS priorities for the transmissions over its adjacent outgoing links (denoted by O_v). To do that, the following simple rules are used:

Rule 6.4.1 (Slot and Priority Selection Rule).

(i) Successful transmissions: *The time-slots at which successful transmissions were realized*

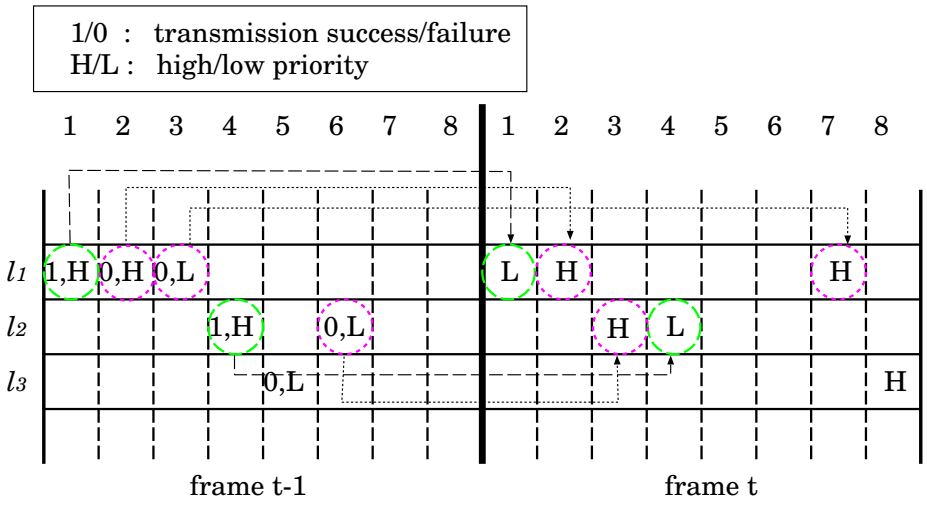


Figure 6.5: Example of Determining Slot-Schedules

at the previous frame are sustained with low control message priority at the current frame.

(ii) Unsuccessful transmissions: *If more time-slots are required (i.e., for transmissions that were not successful at the previous frame), then they are selected at random among the remaining time-slots, with high control message priority.*

To illustrate, consider the example in Figure 6.5, where $O_v = \{l_1, l_2, l_3\}$ with $\rho_{l_1} = 3$, $\rho_{l_2} = 2$, $\rho_{l_3} = 1$, and the frame size is 8. Since at frame $t-1$, the transmission over l_1 on time-slot '1' was successful, this transmission is scheduled once again with *low* control message priority at the same time-slot positions at frame t. The same principle is applied to the transmission over l_2 on time-slot '4.' For the unsuccessful transmissions over l_1 on time-slots '2' and '3', we randomly choose two time-slots of the remaining time-slots, which was not "reserved" by the successful transmissions (i.e., v does not consider time-slots '1' and '4' in this random selection). In the example, time-slot '2' and '7' are selected, and they are scheduled with *high* control message priority from Rule 6.4.1(ii). The same rule is applied to other unsuccessful transmissions. Note that a slot where an unsuccessful transmission was realized at frame $t-1$ could be again scheduled at frame t (e.g., the transmission over l_1 on time-slot '2' at frame t).

Observe that Rule 6.4.1 satisfies the Property 6.4.1. It basically says that any transmission scheduled at some slot s could be re-scheduled with positive probability. In particular,

for a successful transmission, probability that the same time-slot is chosen is '1.' We will use this property in the proof of convergence with the DCAMA algorithm in Section 6.5.

Property 6.4.1. *For any time-slot s, and link l, there exists a positive probability that $c_{ls}[t-1] = c_{ls}[t]$, irrespective of $c_{l's'}[t-1], l' \neq l, s' \neq s$.*

Resolving Contentions

Following the determined slot-schedules, at each time-slot, nodes use the following two-stage RTS/CTS signaling mechanism to resolve contentions and transmit data. Only transmitters having successful RTS/CTS signaling with their receivers are allowed to transmit data.

Two-Stage RTS/CTS Signaling Mechanism

Stage 1: The transmitters and the receivers of high priority transmissions perform RTS-H/CTS-H signaling.

Stage 2: Depending on monitoring status at stage 1, the transmitters and the receivers of transmissions with low priority, which is not forced to release this time-slot by Rule 6.4.2, perform their RTS-L/CTS-L signaling.

Why is signaling in absence of priority inappropriate? First, we explain that signaling without priority could reach a dead-lock condition (i.e., it could stay at an infeasible schedule forever, even if the offered load is feasible). Consider the example in Figure 6.6(a). At frame '0', the transmission over the link A→B on time-slot '1' is unsuccessful due to a collision of RTS messages from A and C at node B, whereas the transmission over link C→E is successful. At frame '1', as we discussed in Section 6.4.2, successful transmissions will be sustained on the same time-slot. Note that any choice of either time-slot '1' or '2' over the link A→B results in unsuccessful transmission due to once again an RTS collision at node B. Thus, even if the offered load is feasible (thus, there exists a feasible schedule), an incorrect choice of initial schedule leads to a deadlock condition.

How does two priority level signaling help? However, if there are two priority levels for control signaling, we can avoid such deadlocks. The reason why we have a deadlock condition with signaling in absence of priority is that there exists a deterministic

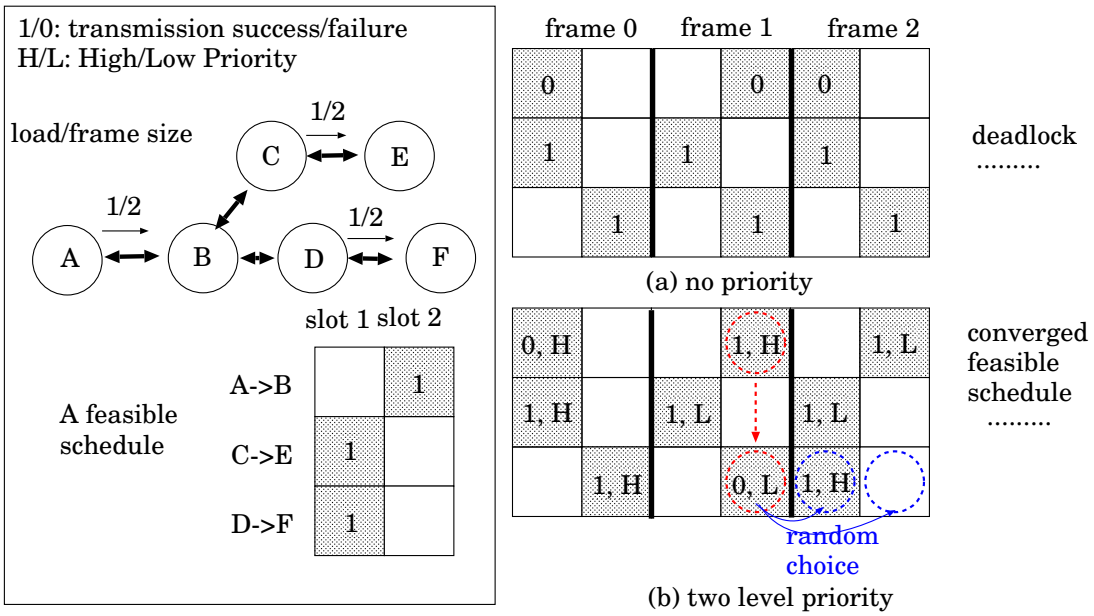


Figure 6.6: RTS/CTS Signaling with and without Priority

"winner-loser" relationship between links, such that if winners maintain their time-slots, no time-slots are available for the transmission by losers. For example, in Figure 6.6, the transmission over the link C→E always wins over link A→B, when both of them are scheduled on the same time-slot, if we use RTS/CTS signaling without priority.

To avoid this deadlock situation, link scheduling algorithms must have a mechanism, whereby there are no deterministic winner-loser relationships. In the DCAMA algorithm, this is achieved by a two priority level of RTS/CTS mechanism, i.e., a scheduled transmission with high priority at some link could "beat" a transmission with low priority at some other link (if those two transmissions have contention relationship as shown in Figures 6.2 and 6.3). The two level priority scheme enables an unsuccessful transmission to preempt a successful one by contending for the channel with high priority signaling. At the same time previously successful transmissions must contend using low priority RTS-L/CTS-L allowing, if need be, possible release of time-slots. This intuition is realized in Stage 2 of two-stage RTS/CTS signaling mechanism, where low priority transmission releases its time-slot (i.e., defers its transmission) by monitoring high priority signaling messages at Stage 1 and applying Time-Slot Release Rule, which will be explained next.

Time-slot release rule. We use $s(l)$ and $d(l)$ to refer to the source and destination of a link l, respectively. We say that a node *senses* a control message if it decodes a control message or receives non-decodable packet collision. Recall that in Section 6.3.1, we assumed that a node can distinguish between the absence of any transmission and packet collisions. We say that a (low priority) transmission over link l *releases* a time-slot s if $s(l)$ or $d(l)$ does not perform RTS/CTS signaling, but the transmission is scheduled on slot s. A low priority transmission decides on its time-slot release (for conflicting high priority transmissions) by conforming to the following simple rule:

Rule 6.4.2 (Time-slot release rule). *A low priority transmission on a given slot s over link l releases the slot s, if on slot s, (i) $s(l)$ senses CTS-H, (ii) $d(l)$ senses RTS-H, (iii) $s(l)$ transmits CTS-H, or (iv) $d(l)$ transmits RTS-H.*

By applying Rule 6.4.2 to low priority transmissions (which will be active at Stage 2), we can easily show that if an high priority transmission has conflicts with a low priority transmission, and both of them are scheduled on the time-slot s, then the high priority transmission makes the low priority transmission release the slot s (see Figures 6.7(a)-(e) for the simple examples). We use "senses" (not "decodes") in Rules 6.4.2(i) and (ii), as there are some cases when the low priority transmission has to release, even if the high priority signaling message is not decodable (see Figure 6.7(g) for an example). Rules 6.4.2(iii) and (iv) comes from half-duplex device constraint (see Figures 6.7(c) and (d)).

Note that the destination of a low priority transmission ($d(l)$) is oblivious to its identity as a receiver before it receives and decodes the corresponding RTS-L message intended for itself. Thus, Rule 6.4.2 (ii) seems to be non-sense. However, if the corresponding RTS-L message is not correctly received (due to packet collisions among low priority transmissions) at Stage 2, $d(l)$ will not send CTS-L message, leading to automatic slot-release.

Further, we reiterate that one of major advantage of *synchronous contention* in Rules 6.4.2(i) and (ii) is that a node is able to identify the kind of control message, irrespective of its decodability, which helps low priority transmission decide on its time-slot release.

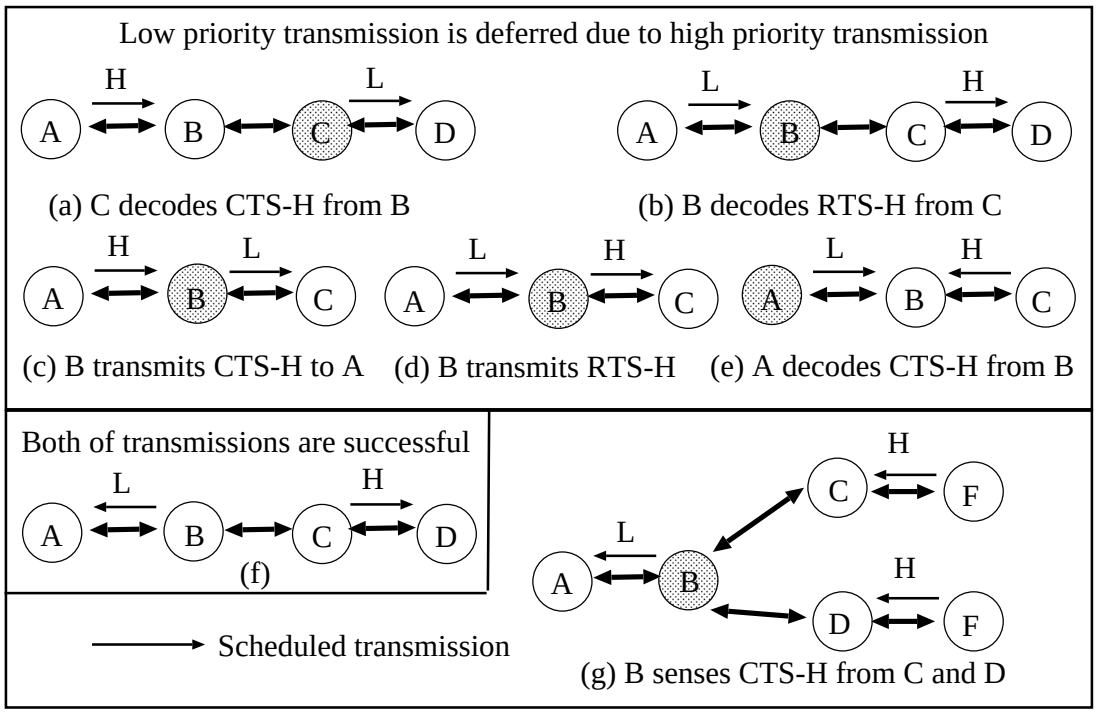


Figure 6.7: Examples for Synchronous Two-Level Priority

6.5 Convergence Results

In this section, we prove that for any feasible offered load, the DCAMA algorithm converges to a feasible schedule exponentially fast. Throughout this section, we implicitly assume that the given offered load is feasible and is denoted by $\vec{\rho}$, and the frame size is F.

First, we provide the Lemma 6.5.1, which is the key to the proof of convergence of the DCAMA algorithm.

For a given feasible offered load and frame size, choose any feasible contention matrix $C^\star$. Let $C[t]$ denote the contention matrix at frame t. We further let $\mathcal{L}'_s[t]$ be the set of links, each of which has a scheduled transmission on time-slot s by $C[t]$ but *not* by $C^\star$ on this time-slot s, i.e.,

$$\mathcal{L}'_s[t] \quad = \quad \{l \in \mathcal{L} \mid c_{ls}[t] = 1,\ c^\star_{ls} = 0\}.$$

Then we have the following result:

Lemma 6.5.1. *Suppose that we have the following conditions at frame t:* (i) *the transmissions over a link $\bar{l}$ is* unsuccessful *on slot s, but it is scheduled on the same slot s by $C^\star$ (i.e., $c_{\bar{l}s}[t] = 1$, $c^\star_{\bar{l}s} = 1$), and* (ii) *all the transmissions over $\mathcal{L}'_s[t]$ at the slot s are* successful.

Then, there is a positive probability that there exists a link $l' \in \mathcal{L}'_s[t]$, such that (i) $C[t+1] = C[t]$, *and* (ii) *the transmission over the link l' becomes* unsuccessful *at frame $t+1$.*

Heuristically, Lemma 6.5.1 states that if a "bad" transmission is successful over l' on slot s (i.e., the feasible schedule $C^\star$ does not schedule it on slot s), and some other link $\bar{l}$ "needs" this time-slot with "good" position (i.e., $C^\star$ has scheduled $\bar{l}$ on slot s), then there is a positively probability that l' will fail in the next frame, leading to the possible movement to one of other time-slots than s at frame $t+2$. This is useful because in the next frame, $\bar{l}$ can possibly claim the slot s from l', and leads towards convergence to $C^\star$. This lemma crucially depends on the *prioritized two level RTS/CTS mechanism* which enables $\bar{l}$ to "beat" l' in the next frame for a good time-slot. The proof is presented in Appendix.

Next, prior to describing the main theorem, we first define a "distance" function between two contention matrices, where distance represents how many different slot-schedules they have between two contention matrices.

Definition 6.5.1. *With a same network topology, a load, and a frame size, consider two contention matrices, $C = (c_{ls})$ and $B = (b_{ls})$. We define*

$$D(C,B) \quad = \quad \sum_{l=1}^{|\mathcal{L}|} \rho_l - \sum_{l=1}^{|\mathcal{L}|} \sum_{s=1}^{F} c_{ls} \times b_{ls}.$$

Intuitively, $D(C,B)$ corresponds to the number of scheduled transmissions by C, each of which is not scheduled by B.

It can be easily shown that if $D(C,B) = 0$, then two contention matrices B and C are equivalent. Thus, for a feasible contention matrix $C^\star$, the fact that $D(C,C^\star) = 0$ implies that C is also feasible.

Theorem 6.5.1 (Convergence). *For an arbitrary graph $\mathcal{G}(\mathcal{L},\mathcal{V})$ with a feasible load $\vec{\rho}$ over the frame-size F, the DCAMA algorithm converges to a feasible contention matrix (i.e., throughput-optimal).*

We first represent the system status at the frame t via $(C[t].R[t])$. We say that a control message priority matrix, $R = (r_{ls})$, is said to *low* if all the scheduled transmissions have low control message priority, i.e., $r_{ls} = 0$ whenever $r_{ls} \neq NULL, \forall s \in \{1, \ldots, F\}, \forall l \in \mathcal{L}$. It can be easily seen that $\{(C[t], R[t]),\ t \geq 0\}$ forms a Markov chain with at least one absorbing state, where an absorbing state corresponds to (C', R') for some feasible C', *low* R'. Note that any state $(C[t], R[t])$, where $C[t]$ is feasible and $R[t]$ is *not* low, goes to an absorbing state over one frame with probability '1' (i.e., $C[t+1], R[t+1]$ will become an absorbing state with probability '1') since $C[t]$'s feasibility ensures that all scheduled transmissions will be successful at frame $t+1$, and all their priorities will be low. Thus, to prove the main theorem, it suffices to show that there is a positive probability that from any initial state, we reach a feasible contention matrix within a finite time.

Our strategy to prove the theorem is that for any fixed feasible contention matrix $C^\star$, we will show that over (at most) two frames there is a positive probability that we get "closer" to $C^\star$ (i.e., $D(C[t+2], C^\star) = D(C[t], C^\star) - 1$), or $C[t+2]$ equals to some other feasible contention matrix $C^{\star\star}, C^{\star\star} \neq C^\star$ (as the feasible contention matrix is not necessarily unique). Since $D(C, C^\star)$ is upper-bounded by $\sum_{l=1}^{|\mathcal{L}|} \rho_l$, for any initial contention matrix C, strict decrease over two frames suffices to prove the convergence. In the proof, we will construct a converging path to a $C^\star$. The proof is presented in Appendix.

Now, we will show that the DCAMA algorithm converges to a feasible contention matrix *exponentially fast*. We first define a random variable $\tau(C)$, corresponding to a convergence time to a feasible contention matrix for a given initial contention matrix C. Then, we have the following exponential rate of convergence.

Theorem 6.5.2 (Rate of Convergence). *For any initial contention matrix C, $\forall t \in \mathcal{Z}_+$, we have*

$$Pr\Big\{\tau(C) > tK\Big\} \quad \leq \quad p^t,$$

for some constants $0 < K < \infty$, and $0 < p < 1$.

The proof is presented in Appendix.

6.6 Adaptive DCAMA

6.6.1 Adaptive Time-slot Access Probability

In the previous section, we have proved that for any feasible load and any network topology, the DCAMA algorithm converges exponentially fast to a feasible schedule.

Note that the DCAMA algorithm chooses a new time-slot (for an unsuccessful transmission) with equal probability in the subsequent frame. In fact, one can potentially increase the rate of convergence or adapt to load change more effectively by intelligently guessing which time-slot is likely to be successful (using the past history), and biasing the time-slot access probabilities. As shown in Proposition 6.6.1 below, such variations of the DCAMA algorithm will also converge to a feasible schedule exponentially fast.

In this section, we propose a general family of variations of DCAMA algorithm, the ADCAMA (Adaptive DCAMA) family, which adaptively assigns different time-slot access probabilities, depending on the past contention history, i.e., more efficient learning of local contention patterns. To that end, each link is assigned its own slot weight vector, and the individual nodes maintain slot weight vectors for its adjacent outgoing links. This slot weight vector is updated every frame by the associated node, depending on the transmission results (success or failure) at the past frames, or overhearing signaling messages around it. Let us denote the slot weight vector of link l at frame t by $\vec{w}^l[t] = (w_s^l[t] : \ s = 1. \cdots, F,)$.

To increase/decrease the slot weight vector based on the past contention histories, we define the *time-slot status*, which corresponds to the result of past contentions (e.g., success or failure) on the corresponding time-slots. Then, the slot access probability is set to be *inversely proportional* to the current weight. Also, by setting the minimum and maximum of weight, we can avoid pathological cases (e.g., the time-slot access probability could be arbitrarily small or close to '1'), i.e., there exist $\bar{w}$ and $\underline{w}$, such that $1 \leq \underline{w} < \bar{w} < \infty$ and $\forall s \in \{1, 2, \ldots, F\}, \forall l \in \mathcal{L}$, and $\forall t > 0$, $\underline{w} \leq w_s^l[t] \leq \bar{w}$.

Then, we define a m-frame history based ADCAMA algorithm, where each node stores and uses the previous m-frame slot status history, based on which slot weight vector is updated at every frame. Intuition behind the multi-frame history based algorithm is

that we could potentially increase the rate of convergence or have the higher transient throughput by considering longer slot usage history. As an example, a time-slot with consecutive success is highly likely to be "safe", so that it would be beneficial to sustain the corresponding time-slot at the next frame[5]. In Section 6.7, we will show that even with a simple weight maintenance algorithm based on three frame contention history, we could have quite a performance increase, compared with the DCAMA algorithm.

6.6.2 Convergence Results

Now, we have the following proposition to Theorem 6.5.1:

Proposition 6.6.1 (Convergence of ADCAMA). *For an arbitrary graph $\mathcal{G}(\mathcal{L}, \mathcal{V})$ with a feasible load $\vec{\rho}$ over the frame-size F, any m-frame history based ADCAMA algorithm converges (exponentially fast) to a feasible contention matrix (i.e., throughput-optimal).*

Proof. Let us define a state at frame t by

$$\mathbf{X}_m[t] \triangleq \Big((C[t-m+1], R[t-m+1]), \cdots, (C[t], R[t])\Big),$$

where $C[n] = R[n] = 0$, for $n < 0$. Then, it is clear that $\{\mathbf{X}_m[t], t \geq 0\}$ forms a Markov chain with at least one absorbing state, where $\mathbf{X}_m[t_\star]$ for some frame $t_\star$, is an absorbing state if $\forall i = 0, \cdots, m-1$, $C[t_\star - i]$ is feasible, and $R[t_\star - i]$ is low. We observe that if $\mathbf{X}_m[t_0]$ is not an absorbing state, but $C[t_0]$ is feasible for some frame t_0, then $\mathbf{X}_m[t_0]$ goes to an absorbing state over *at most* m-steps (m-frames) with probability '1'. Thus, it suffices to show that there is a positive probability that from any initial state, we reach a feasible contention matrix within a finite time.

The only difference between the DCAMA and the ADCAMA is that they use different time-slot access probabilities. However, the slot access probability with ADCAMA are still guaranteed to be strictly positive. Thus, the proof is similar to that in Theorem 6.5.1. □

[5]Thus, DCAMA algorithm corresponds an algorithm belonging to the ADCAMA family. However, we use the term 'DCAMA' to refer to an ADCAMA algorithm without frame history

Table 6.1: Parameters Used for Weight Increase/Decrease

$S_s^l[t-3]$	$S_s^l[t-2]$	$S_s^l[t-1]$	inc/dec Weight
SUCC	SUCC	SUCC	$-D_1$
FAIL/IDLE	SUCC	SUCC	$-D_2$
FAIL	FAIL	FAIL	$+I_1$
SUCC/IDLE	FAIL	FAIL	$+I_2$

6.7 Simulation Results

In this section, we simulate wireless multi-hop networks with nodes which are randomly distributed in a 500×500 or 100×100 meter-square area. The number of nodes, their transmission range, and the frame size are parameterized, such that we can observe the performance of the proposed algorithms under different connectivity densities, time varying environments and MAC layer rate granularities.

6.7.1 Weight Maintenance Algorithm

In Section 6.6, we have proposed a family of DCAMA variations (ADCAMA) adaptively assigning different time-slot access probabilities. We now describe the details of a simple weight maintenance strategy based on three-frame history. To summarize our strategy, we increase/decrease slot weights (equivalently, the slot access probabilities, see Section 6.6.1) based on observed success/failure of past time-slot requests. We show that even with a simple weight update mechanism, the performance of DCAMA can be improved significantly, and enables it to be more adaptive to load/topology changes.

We denote a slot status over link l at time-slot s at frame t by $S_s^l[t]$. We have three kinds of time-slot status: success (SUCC), failure (FAIL), and idle (IDLE). The IDLE status corresponds to the case when a node which did not sense any control message.

Table 6.1 shows the parameters used in the simulation for a typical link l. The parameters I_i and D_i are the (additive) weight increase/decrease constants used by nodes to adapt their slot weights based on past observations. Table 6.1 summarizes the observed state over the past three frames, and the corresponding weight change operation. These parameters are chosen such that $D_1 > D_2 > 0$, and $I_2 > I_1 > 0$. We have used $D_1 = I_1 = 3$,

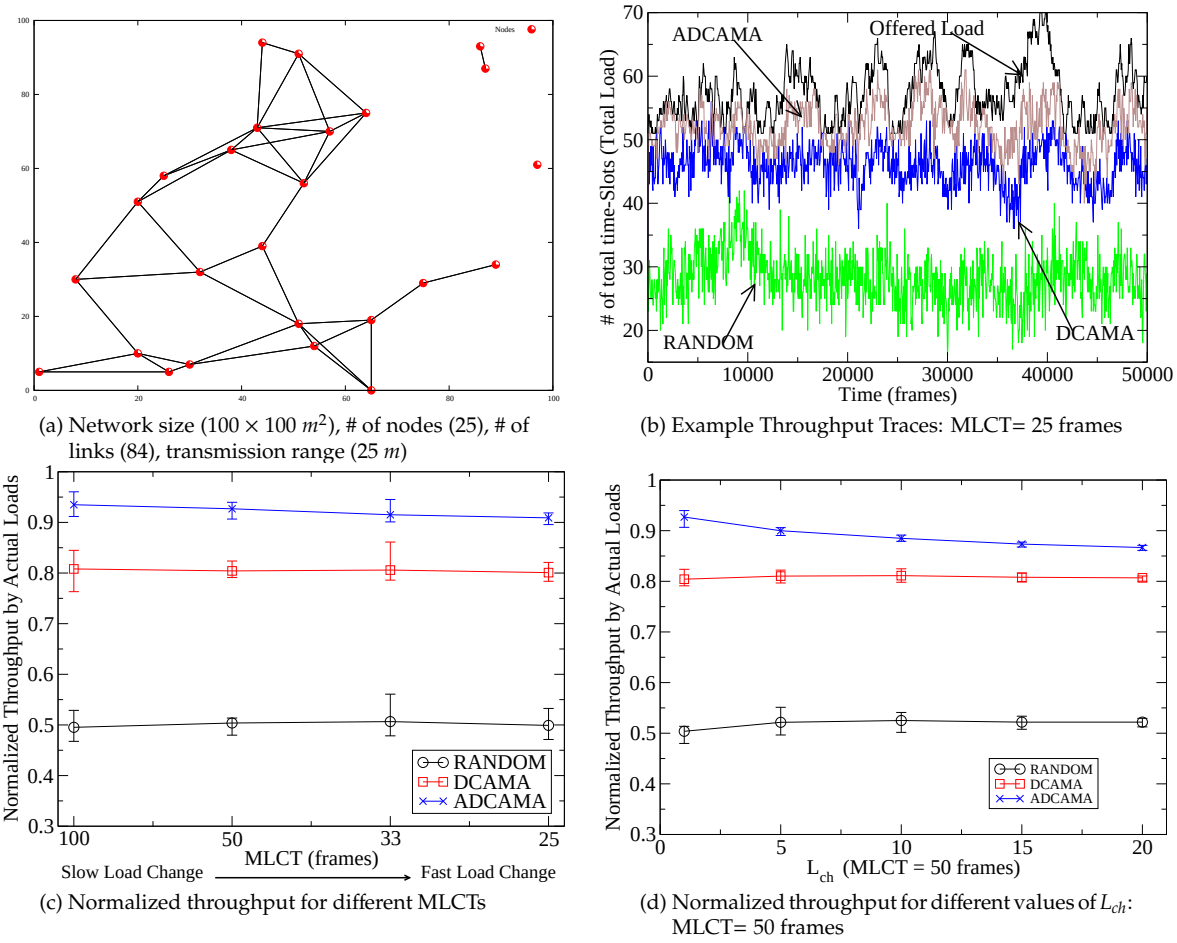
$D_2 = I_2 = 1$, in all simulation results, where the maximum and minimum weights (i.e., $\bar{w}$ and $\underline{w}$) are set to 30 and 1, respectively (recall that the time-slot access probabilities are inversely proportional to weights).

The intuition for these choices is that more back-to-back successes at a time slot indicate that the offered loads around the corresponding node at that time-slot are relatively low (i.e., less "congested"), and transmissions in that time-slot are likely to be successful in the future. Similar intuition is applied for back-to-back failures. However, empirical evidence based on simulations have indicated that responding to just a one-time success/failure by decreasing/increasing the weight was not very helpful, because such a success/failure could have happened due to transient movement of transmission schedules at other conflicting links (i.e., it does not capture congestion very well).

With regard to the IDLE status, it seems intuitive to schedule an unsuccessful transmission at the IDLE time-slot with higher probability (i.e., decrease the weights) in order to to "spread" the offered load over all the time-slots of a frame. However, weight decrease at the IDLE status could generate a synchronization effect, i.e., due to aggressive decrease (by multiple nodes), multiple transmissions are highly likely to be scheduled at this slot, leading to collision again. Based on empirical evidence using simulations, responding aggressively to SUCC and FAIL is the determining factor in providing fast convergence and good adaptability. We finally comment that using other numerical values for D_j, I_j based on the heuristics above also results in significant performance improvements (compared to DCAMA), thus indicating that these heuristics are quite robust to the actual numerical values. We do not present simulations for varying D_j, I_j due to space constraints.

6.7.2 Simulation Results

In this section, we evaluate the performance of DCAMA and ADCAMA algorithm by comparing them to the RANDOM algorithm, described below. The RANDOM algorithm determines slot-schedules (based on the requested loads) in a pure-random manner at each frame, and uses a single-level RTS/CTS signaling to gain access to the channel. The reason why we adopt the RANDOM algorithm as a base-line comparison is because *(i)* it is similar to Aloha-like strategy (which is a "standard" algorithm for TDMA link scheduling), and



(a) Network size (100×100 m^2), # of nodes (25), # of links (84), transmission range (25 m)

(b) Example Throughput Traces: MLCT= 25 frames

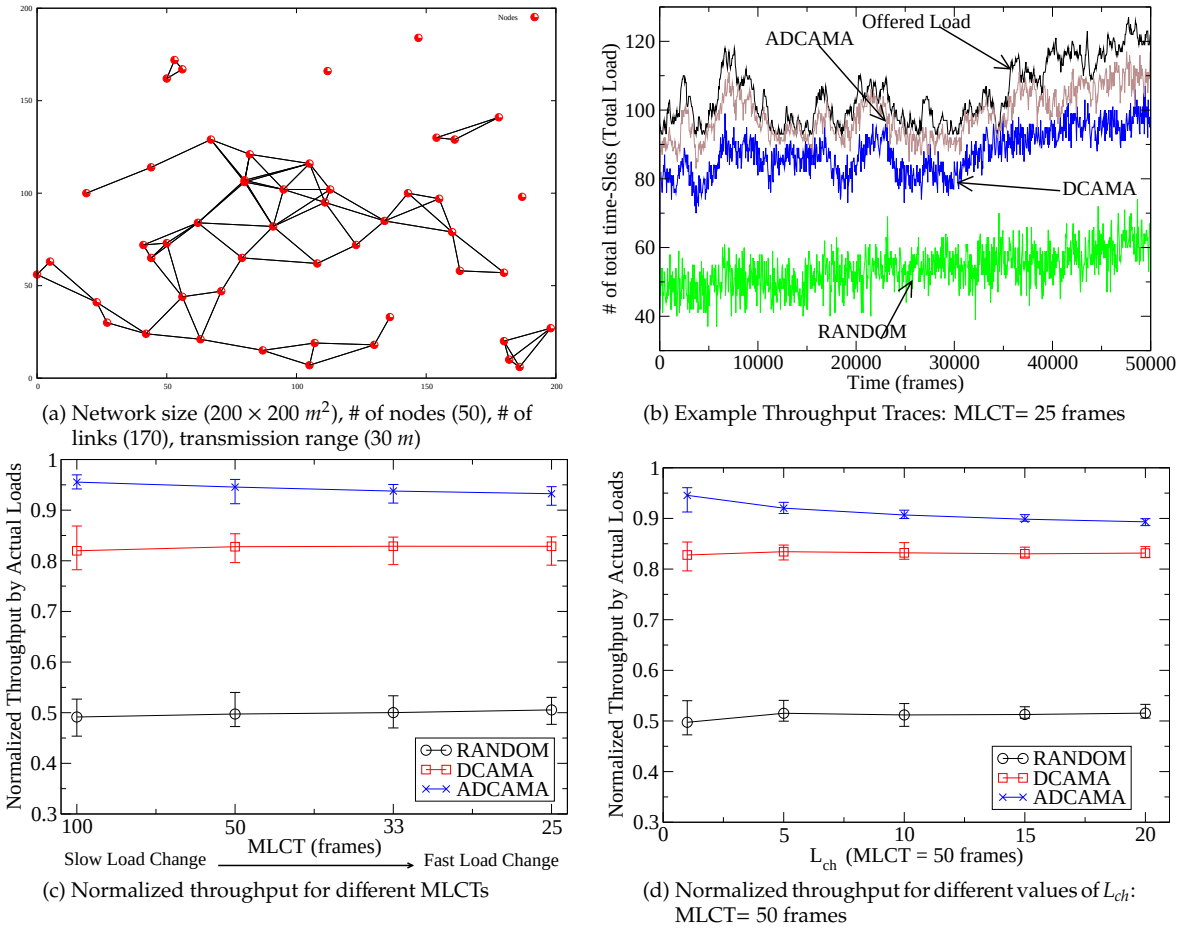
(c) Normalized throughput for different MLCTs

(d) Normalized throughput for different values of L_{ch}: MLCT= 50 frames

Figure 6.8: With frame size of 10 and the network topology (a), (b) shows an example traces of # of time-slots with successful transmissions, compared to the actual loads. (c) and (d) show the normalized throughput w.r.t the actual loads over 50000 frames for different values of MLCTs and L_{ch}.

behaves like a slotted version of a CSMA-like contention-based scheme, and *(ii)* it is not clear how we can compare with some of the other dynamic coloring based algorithms, since their objective is to solve a variant of the coloring problem with different system models (such as two-hop control message exchange and different transmission conflict scenarios, see Section 6.2).

Prior to presenting simulation results, we comment on the control overhead of the DCAMA/ADCAMA algorithm. Our approach has additional overheads as compared to a standard contention based MAC protocol (which has only one RTS/CTS signaling phase).



(a) Network size (200 × 200 m^2), # of nodes (50), # of links (170), transmission range (30 m)

(b) Example Throughput Traces: MLCT= 25 frames

(c) Normalized throughput for different MLCTs

(d) Normalized throughput for different values of L_{ch}: MLCT= 50 frames

Figure 6.9: The analogous simulation results to those in Figure 6.8 but for the different network topology (a).

Suppose that a MAC packet has 1000 bytes of data (note that in the 802.11 MAC, the size limit is 2312 bytes). The overhead of each RTS/CTS message pair with DCAMA is no more than 30 bytes (6 bytes each for source/destination addresses, and 3 bytes for signaling such as RTS priority level, stage, etc) will suffice for our protocol. Thus, the additional overhead (recall that the "standard" protocol has only one stage of RTS/CTS messages) is about 30 bytes, which corresponds to approximately 3%.

On the other-hand, we have significant throughput gains when compared to a baseline random access MAC, and our simulations indicate a 25-30% gain in various scenarios (changing topology, load requirements. steady-state, etc). Thus, it seems worthwhile to

pay the penalty of additional overheads in order to accrue this additional throughput gain. Another overhead with a fixed slot-size based approach is due to partial-slot wastage, i.e., a small packet in one time slot wastes part of a time-slot and thus reduces time resource usage. However, this could be overcome by using "packet bursting" or "packet aggregation" [122], where a time-slot usage is maximized by aggregating the packets intelligently.

There are additional issues in comparing the DCAMA/ADCAMA algorithm with conventional "static" TDMA algorithms (please see Section 6.2 for the related work). In a static TDMA algorithm, with every load/topology change, the scheduling decision has to re-computed, for which control messages has to be exchanged, and most of the research in literature assumes that the control messages are successfully transfered to neighboring nodes contention-free. However, in a single channel wireless ad-hoc network, this assumption seems to be unrealistic. Thus, it may take some time to disseminate and share the newly generated scheduling decision. On the other hand, our approach does not make any such assumptions, and indeed RTS/CTS collisions could occur, leading to control message losses.

adaptability to load changes. First, we investigate the effect of load changes on the performance of DCAMA and ADCAMA algorithm with frame size of 10 in the network topology of Figure 6.8(a). We generate time-varying loads by a random walk model, where we first determine a normalized offered load of 70% (by a randomly chosen maximally feasible load[6]). Then, at the beginning of each frame we randomly choose L_{ch} links and increase their link loads by one slot with probability P_L^I, decrease their link loads with probability P_L^D, or stay at the current load (i.e., no change) with probability $1 - P_L^I - P_L^D$. For simplicity, in the simulation, we set $P_L \triangleq P_L^I = P_L^D$. Thus, higher values of P_L corresponds to a faster load change with time. Then, the mean load change time (MLCT) over L_{ch} links is $1/(2 \times P_L)$ frames.

Figure 6.8(c) shows that the throughput (over 50000 frames) normalized by the actual (time-varying) offered load for different values of MLCTs ($L_{ch} = 1$) varying from 25 to 100 frames, where the error bars represent the maximum and minimum values of 10 simulations with different random seed values (i.e., different load changing patterns). For a network

[6] A load is said to be *maximally feasible* if the resulting system load becomes infeasible with *any* load increase anywhere in the network.

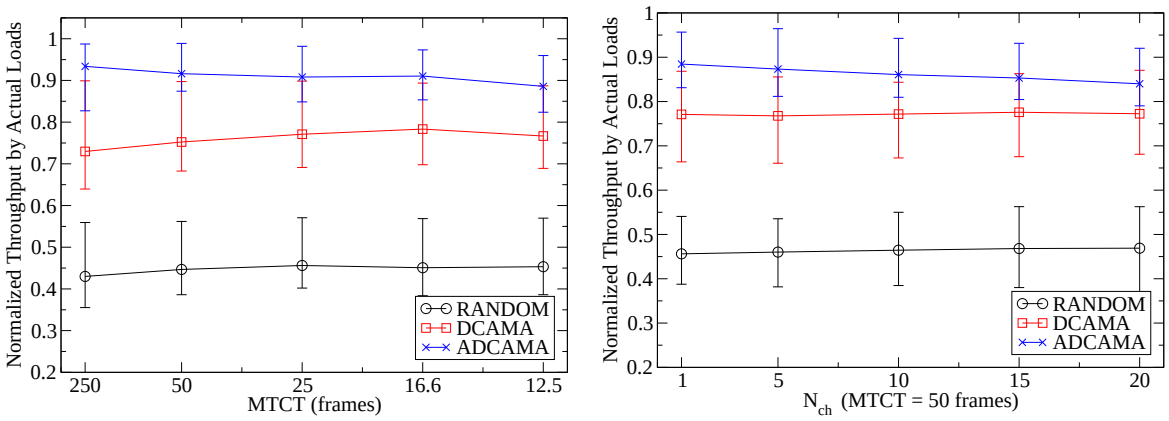
with a link capacity of 10 Mbps, and a frame-size of 10 (which corresponds to a 10 msec frame duration), this corresponds to a load change ranging from once every 250 msec to once every 1 seconds.

We observe that with ADCAMA algorithm, the normalized throughput is above 90%, whereas the RANDOM achieves about 50%. Figure 6.8(b) shows an example trace of throughput (i.e., number of successful transmission slots) for MLCT= 25 frames, where we observe that ADCAMA algorithm tracks the actual load very well, resulting in nice adaptability to time-varying load changes.

Figure 6.8(d) shows the normalized throughput by actual offered loads in faster load changing scenario, where with MLCT= 50 frames, L_{ch} varies from 1 to 20. Note that the actual mean load change time for L_{ch} = 20 is 50/20 = 2.5 frames, which corresponds to 25 msec. As L_{ch} becomes larger, the throughput difference between DCAMA and ADCAMA becomes slightly smaller. This is because with faster changing loads, ADCAMA algorithm does not have sufficient time to completely adapt to changes. However, even in this fast changing regime, ADCAMA shows a 5% throughput improvement over DCAMA.

Figure 6.9 shows the analogous simulation results to those in Figure 6.8 for a different network topology.

Adaptability to topology change. Second, we investigate the effect of topology changes on the performance of DCAMA and ADCAMA algorithm. With time-varying topology changes, new links and existing links are dynamically added and deleted in the network, which possibly changes the offered loads in the links. To simulate practical scenarios of such load/topology changes, we use end-to-end flows to generate offered loads in the network. For a fixed frame size, we first randomly generate a network topology. To initialize the end-to-end flows, we employ the following procedure: we randomly choose two nodes (source and destination of an end-to-end connection), and increase the load in the path of the chosen source-destination end-to-end connection. We assume that a shortest path routing is used to determine the path. We carry out this random selection of end-to-end connections $2 * |\mathcal{V}|$ times (note that $|\mathcal{V}|$ is the number of nodes in the network), and skip the chosen end-to-end connection if the load increase of its path generates infeasible offered load. Thus, the initially generated loads are very close to a maximally feasible load.



(a) Normalized throughput for different MTCTs

(b) Normalized throughput for different values of N_{ch}: MTCT= 50 frames

Figure 6.10: With 100×100 m^2 network size, 30 nodes, 25m transmission range, and frame size of 10, we first uniformly place nodes in the plane, and generate end-to-end connections randomly. (a) and (b) show the normalized throughput w.r.t the actual loads for different values of MTCTs and N_{ch}.

For the evolving time-varying topology, we randomly choose N_{ch} nodes and move each of the nodes independently in one of four directions (north, south, east, or west) by one meter with probabilities of P_T^N, P_T^S, P_T^E, or P_T^W. In the simulations, we set them to be all equal, denoted by P_T. Then, the mean topology change time (MTCT) is $1/(4 \times P_T)$ frames. If the topology changes due to node movement, we establish the new end-to-end routes based on the shortest path routing, for initially found end-to-end connections.

Figure 6.10(a) shows that the throughput (over 10000 frames) normalized by the actual (time-varying) offered load (due to topology changes) for different values of MTCTs varying from 12.5 frames to 250 frames. Again, for a network with a link capacity of 10 Mbps with the frame size of 10, this corresponds to a change ranging from once every 125 msec to once every 2.5 seconds, which corresponds to a mobile terminal moving with an average velocity of 8 m/sec and 0.4 m/sec. This seems reasonable for a typical slowly varying environment. Similar to the figures in load change simulations, we represent the maximum and the minimum values of error bars in 10 simulations with different random seed values, leading to different initial network topologies and end-to-end connections.

Further, Figure 6.10(b) shows the normalized throughput when with MTCT= 25 frames, N_{ch} varies from 1 to 20 nodes. We again observe that with ADCAMA algorithm,

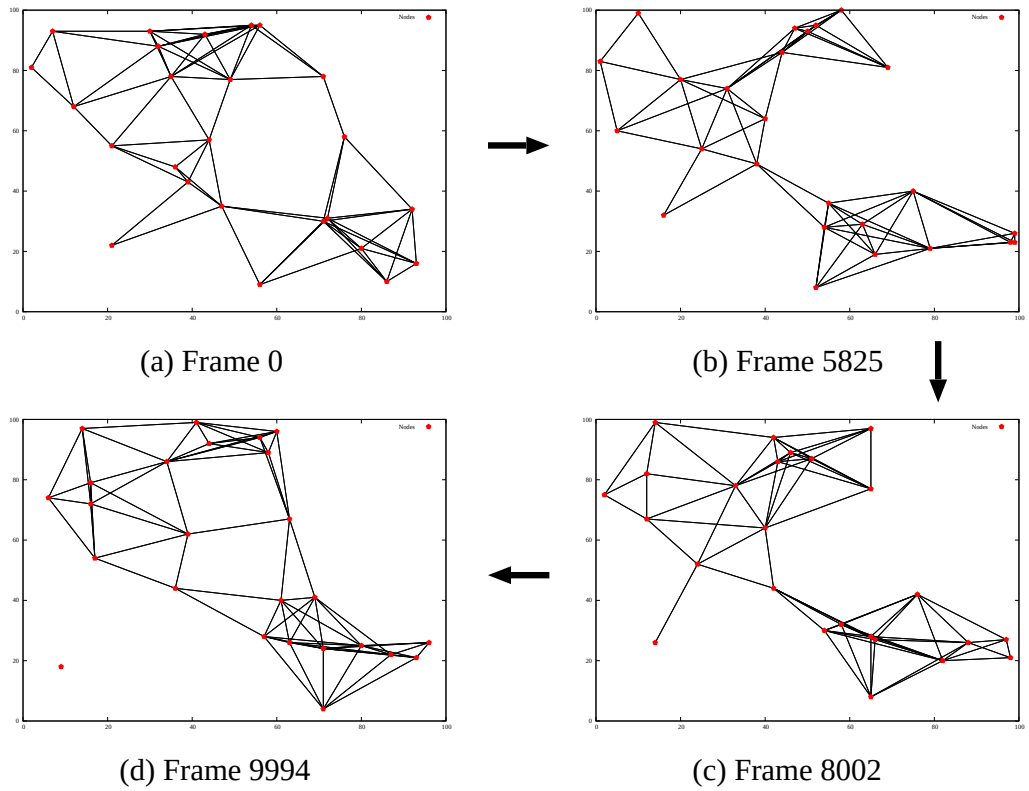
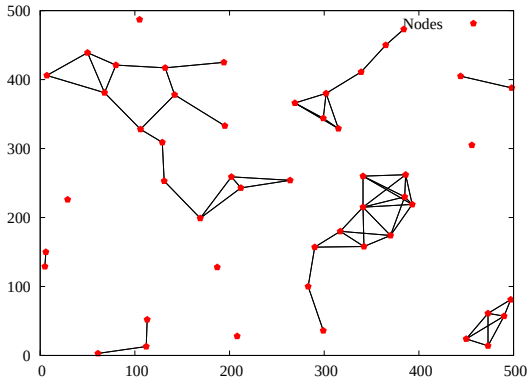


Figure 6.11: This figure shows an instance of topology changes (due to the random-walk model) at four time instances.

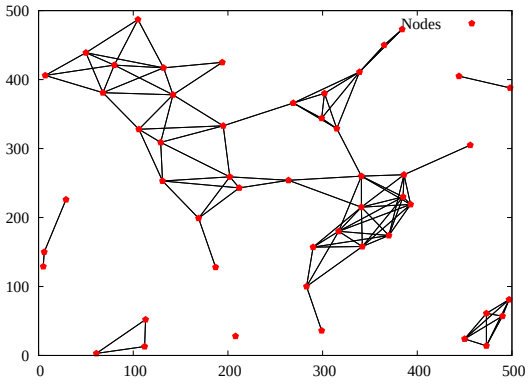
the normalized throughput is above 90%, whereas RANDOM achieves at most 60%. Thus, ADCAMA algorithm exhibits good robustness to topology changes.

Effect of connectivity densities and frame sizes. Next, we investigate the effect of different connectivity densities and frame sizes on the performance of DCAMA/ADCAMA algorithm at the "steady" state (i.e., no load/topology change for some time). Figure 6.12 shows the performance of DCAMA and ADCAMA algorithms for a normalized load (by a randomly chosen maximally feasible load), which varies from 10% to 100%. We measure the aggregate normalized throughput for every varying load over 3000 frames. Each point in the graph is the mean value of 50 simulation experiments with different random seed values.

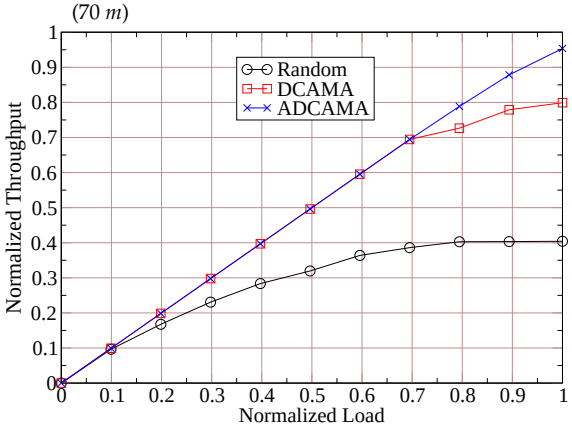
We observe that ADCAMA algorithm has better transient throughput than DCAMA, and exhibits fast convergence. In particular, with ADCAMA algorithm, up to 90% load (for



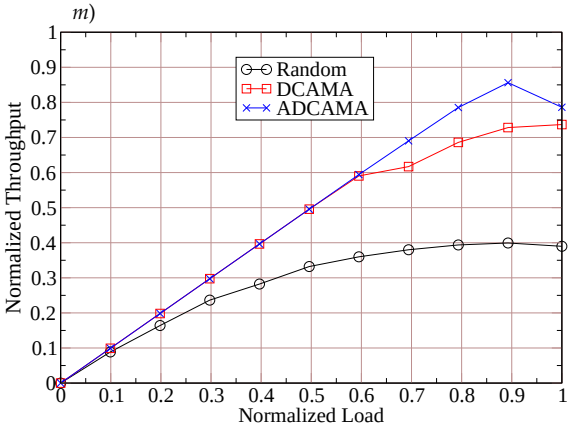
(a) Sparse Network: network size (500×500 m^2), # of nodes (50), # of links (124), transmission range (70 m)



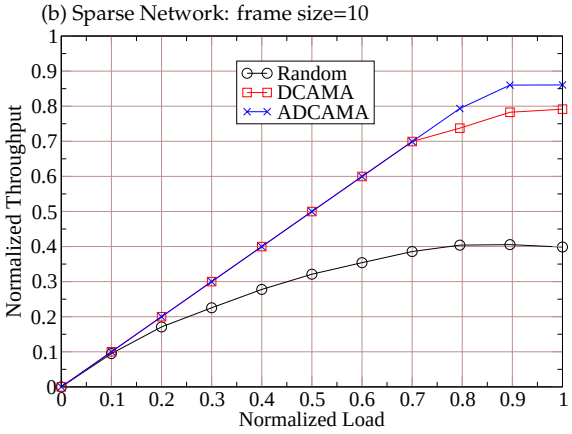
(d) Dense Network: network size (500×500 m^2), # of nodes (50), # of links (206), transmission range (90 m)



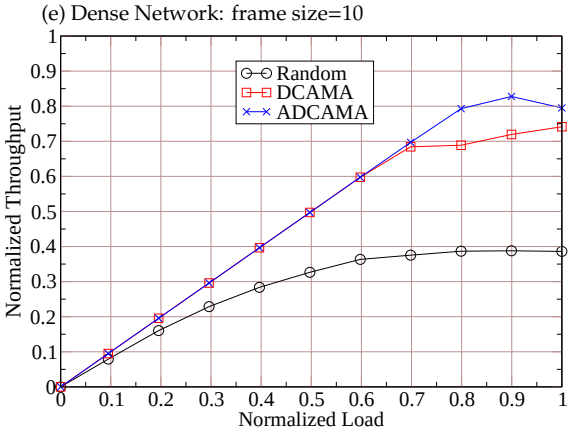
(b) Sparse Network: frame size=10



(e) Dense Network: frame size=10



(c) Sparse Network: frame size=15



(f) Dense Network: frame size=15

Figure 6.12: Throughput for Different Network Connectivities and Frame sizes

frame size of 10) and 80% load (for frame size of 15), convergence will occur very fast. Thus, assigning slot access probabilities *adaptively* by remembering the past contention patterns

is beneficial to achieving high transient throughput and fast convergence to a conflict-free schedule.

We also observe that with increasing frame sizes and network connectivity densities, the aggregate throughput decreases by approximately 5% at high loads. An intuitive interpretation is that as the frame size increases, the number of slots to choose (randomly) for unsuccessful transmission also increases, which causes the system to take longer time to reach a feasible state. Similarly, as network densities increases (i.e., the number of connected links increases), the chance for every transmission to be in conflict-free slots becomes smaller. However, the frame size is usually less than 15 (e.g., in a GPRS network) in order to support bounded worst case delay at the MAC layer. The bounded worst case delay is important to real-time applications and a time-out based transport protocol such as TCP. Thus, frame size seems to be bounded by a small number, in which case ADCAMA algorithm has very good performance. Further, network connectivity density is typically bounded due to the limited number of nodes and the transmission power adjustment for energy-savings.

6.8 Extension to System with Power Control

In this section, we extend our analysis and DCAMA algorithm to a system with power control, where a discrete level of powers is available for transmission. We also show that such an extended algorithm also converges exponentially fast to a feasible for any feasible offered load and any network topology.

6.8.1 System Model with Power Control

First, we describe the extended system model to that in Section 6.3, where there exist discrete power levels being used to transmit both control message and data. We denote the set of such power levels by $\mathcal{P} = \{P_0 = 0, P_1, P_2, \ldots, P_m\}$. The '0' power corresponds to the case when the transmission does not occur. We assume that in a transmission with some power P_i from node v to node w, if w is inside the v's transmission range, then v is also inside the transmission range in a transmission from w to v with the same power P_i. If a

transmission is said to be scheduled with the power P_i, both RTS/CTS messages and data are transmitted with the power P_i.

We represent the power allocation status in the system by a *power matrix* (PM) $P(F, \vec{\rho}) = (p_{ls} : l = 1, \ldots, |\mathcal{L}|, s = 1, \ldots, F)$ for a frame-size F and an offered load $\vec{\rho}$, where $p_{ls} = P_i$ implies that the transmission over over link l on slot s is scheduled with the power P_i. Then, the system status at a frame t is represented by $(C[t], R[t], P[t] : t = 1, 2, \ldots)$.

Now, similar to Section 6.3, we provide extended definitions on the feasibility of an offered load and contention matrix. For a offered load $\vec{\rho}$, and frame size F, a pair of contention-power matrix, $(C(F, \vec{\rho}), P(F, \vec{\rho}))$ is said to be *feasible* if all its links are satisfied. An offered load $\vec{\rho}$ is said to be *feasible* over a frame size F if there exists a feasible $(C(F, \vec{\rho}), P(F, \vec{\rho}))$. A contention matrix $C(F, \vec{\rho})$ is said to be *potentially feasible* if there exists a power matrix $P(F, \vec{\rho})$, such that $(C(F, \vec{\rho}), P(F, \vec{\rho}))$ is feasible.

6.8.2 Algorithm and Convergence

In this section, we describe an algorithm, which is throughput-optimal with exponential rate of convergence. We focus on the per-frame operation (i.e., determining slot schedules, control message priorities, power levels for scheduled transmissions), which is described in Rule 6.8.1. The per-slot operation is equivalent to that in Section 6.4.2, except that different power levels are used for control message and data transmissions.

Rule 6.8.1 (Slot, Priority, and Power Selection Rule).

(i) Successful transmissions: *Both the time-slots and the power levels at which successful transmissions were realized at the previous frame are sustained with low control message priority at the current frame.*

(ii) Unsuccessful transmissions: *If more time-slots are required (i.e., for transmissions that were not successful on the previous frame), then time-slots and power levels are selected at random among the remaining time-slots and* $\mathcal{P} \setminus \{0\}$*, respectively, and their control message priorities are set to be high.*

Next, we discuss the proof of convergence and its rate of convergence. Due to space limitation, we skip the complete proofs. However, as we can see in Rule 6.8.1, the algorithm

is similar to the original DCAMA algorithm except for using randomly chosen power level, leading to analogous proofs in the algorithm with power control.

First, we present the following Lemma 6.8.1:

Lemma 6.8.1. *Consider a contention and power matrix* $(C[t], P[t])$ *at frame* t*, where* $C[t]$ *is potentially feasible. Then, there exists a positive probability that* $(C[t+t_1], P[t+t_1])$ *is feasible for some* $0 \leq t_1 < \infty$*, such that* $C[t+t_1] = C[t]$.

Lemma 6.8.1 can be proved by the following heuristic argument: From Property 6.4.1, there is a positive probability that $C[t+s] = C[t]$, for all $s < \infty$. Further, a successful transmission with "badly large" power allocation, which makes other transmission unsuccessful, can be beaten by the unsuccessful transmission (with enough power being randomly chosen to beat the successful transmission) due to prioritized RTS/CTS signaling. Then, from Lemma 6.8.1, it suffices to show that the system will reach a $(C^\star, P)$, for a potentially feasible contention matrix $C^\star$.

Theorem 6.8.1 (Convergence). *For an arbitrary graph* $\mathcal{G}(\mathcal{L}, \mathcal{V})$ *with a feasible load* $\vec{\rho}$ *over the frame-size* F*, the DCAMA algorithm converges to a feasible* $(C^\star, P^\star)$ *(i.e., throughput-optimal). Further, its rate of convergence is exponential.*

The proof of Theorem 6.8.1 is also quite similar to that of Theorem 6.5.1. The key observation is that there does not exist a deterministic winner-loser relationship between any two simultaneously scheduled transmissions on a same time-slot, i.e., a unsuccessful high priority transmission can preempt a successful conflicting transmission with low priority by using high priority RTS/CTS mechanism and appropriately chosen power at random.

Similar to ADCAMA algorithms, we can develop an adaptive algorithm, which uses the past power usage histories to have adaptive power-level access probabilities. Potentially, this scheme can be incorporated with adaptive time-slot access probability by maintaining power weight vector for link l at frame t, denoted by $\vec{\theta}^l = (\theta^l_i : i = 1, 2, \ldots, m)$. Recall that m is the number of discrete power levels.

6.9 Discussion of Relation with Fluid Models

In Chapter 5, we had modeled the MAC-layer by means of a fluid model, and studied congestion control algorithms using this model. The study on the MAC protocol in this chapter has dealt with the complementary discrete problem of determining dynamic schedulers, such that over a long time-scale, the user rates with the MAC can be approximated using a fluid model. As shown in the simulation results of this chapter, the (A)DCAMA algorithm converges to a throughput-optimal schedule under a moderate load quite quickly (i.e., within about 100 frames). This leads to a natural time-scale separation between congestion control and MAC scheduling (and thus, fluid models of the MAC) as long as the congestion controller adapts slowly.

We finally comment that the system model of the wireless network in Chapter 5 differs from that in this chapter. In Chapter 5, we had considered a *node-exclusive interference model,* i.e., there exist multiple available channel resources, such that resource is allowed to be reused in the network, as long as transmissions does not share a common node. However, in this chapter, we have assumed a single channel wireless system, where two-hop distant transmissions could have a collision depending on their directions (i.e., hidden-node terminal). The congestion control algorithms in Chapter 5 can be readily extended to a single-channel wireless system, and examples of related work includes [86].

Appendix

Proof of Lemma 6.5.1

Proof. We first let $\tilde{\mathcal{L}}_s$ be the set of links, having a scheduled transmission on time-slot s, and it is also scheduled by $C^\star$ on time-slot s i.e.,

$$\tilde{\mathcal{L}}_s[t] \quad = \quad \{l \in \mathcal{L} \mid c_{ls}[t] = 1,\ c^\star_{ls} = 1\}.$$

Note that the set of all links having scheduled transmissions at time-slot s at frame t is $\mathcal{L}'_s[t] \cup \tilde{\mathcal{L}}_s[t]$. Next, let $\bar{\mathcal{L}}_s[t]$ denote be the subset of $\tilde{\mathcal{L}}_s[t]$, such that if $l \in \bar{\mathcal{L}}_s[t]$, then the

transmission over l was *unsuccessful,* at frame t, i.e.,

$$\bar{\mathcal{L}}_s[t] = \{l \in \mathcal{L} \mid c_{ls}[t] = 1,\ c^{\star}_{ls} = 1,$$
$$\text{the transmission over } l \text{ at slot } s \text{ are unsuccessful}\}.$$

For notational simplicity, we henceforth omit the frame time parameter, $[t]$, in $\mathcal{L}'_s[t]$, $\tilde{\mathcal{L}}_s[t]$, and $\bar{\mathcal{L}}_s[t]$, unless explicitly needed.

With this setup, it is clear that $\bar{\mathcal{L}}_s \subset \tilde{\mathcal{L}}_s$, $\tilde{\mathcal{L}}_s \cap \mathcal{L}'_s = \emptyset$, and $\bar{l} \in \bar{\mathcal{L}}_s$, since in the lemma statement, we assumed that the transmission over $\bar{l}$ was unsuccessful.

Now, we introduce the useful concept, *contention group*, to understand contention relationship from graph-theoretic perspective. We define a *link contention graph,* whose vertices correspond to the links in the original graph $\mathcal{G}(\mathcal{L}, \mathcal{V})$. In the link contention graph, two *vertices* $i, j \in \mathcal{L}$ are connected, if at most one transmission is successful when both of them are scheduled on the same time-slot, i.e., two nodes (links in the original graph) in the link contention graph has one of the relationships in Figure 6.2 and Figure 6.3.

Then, we define a contention group as a *maximal clique*[7] of a link contention graph. Thus, for a conflict-free schedule, *at most* one of links in the same contention group has to be scheduled at the same time-slot. Note that the network might have multiple contention groups.

Then, all the links in $\tilde{\mathcal{L}}_s$ (thus, also $\bar{\mathcal{L}}_s$) are in the *different contention groups,* since $C^{\star}$ is feasible and at most one of links in the same contention group has to be scheduled at the same time-slot (in this case, s) in a feasible contention matrix. Further, since there is no collision between the links inside $\tilde{\mathcal{L}}_s$ (due to feasibility of $C^{\star}$), the assumption that the transmission over $\bar{l}$ at slot s is unsuccessful implies that *there exists a link* $l' \in \mathcal{L}'_s$*, such that* $\bar{l}$ *and* l' *shares the same contention group* (i.e., there must be a colliding transmission in $\mathcal{L}'_s$ which makes the transmission over $\bar{l}$ unsuccessful). For the $l' \in \mathcal{L}'_s$, let

$$\bar{\mathcal{L}}^{l'}_s \quad = \quad \{l \in \mathcal{L}'_s \cup \tilde{\mathcal{L}}_s \mid l \text{ shares a contention group with } l'\},$$

[7] A maximal clique is a complete subgraph such that if one further node were included anywhere the completeness condition would be violated.

i.e., $\bar{\mathcal{L}}_s^{l'}$ is the set of links which have a scheduled transmission on slot s and shares a contention group with l'. With this definition, we must have that $\bar{\mathcal{L}}_s^{l'} \subset \bar{\mathcal{L}}_s$, because (i) all the links in $\mathcal{L}_s'$ have successful transmissions from assumption, and (ii) all the links in $\bar{\mathcal{L}}_s^{l'}$ have unsuccessful transmissions, and at most one transmission can be successful within a contention group.

We also note that *all the links in $\bar{\mathcal{L}}_s^{l'}$ are from different contention groups*, since all the links in $\tilde{\mathcal{L}}_s$ ($\supset \bar{\mathcal{L}}_s \supset \bar{\mathcal{L}}_s^{l'}$) are from different contention groups. Further, we note that $\bar{l} \in \bar{\mathcal{L}}_s^{l'}$, i.e., $\bar{\mathcal{L}}_s^{l'} \neq \emptyset$.

From Property 6.4.1, there is a positive probability that we have $C[t+1] = C[t]$, i.e., all the transmissions at frame $t+1$ are again scheduled at the same time-slot positions as at frame t. Then, from Principle 6.4.1, we must have

$$r_{ls}[t+1] = \begin{cases} 1 & \text{if } l \in \bar{\mathcal{L}}_s, \text{ i.e., high priority} \\ 0 & \text{if } l \in \mathcal{L}_s' \cup (\tilde{\mathcal{L}}_s \setminus \bar{\mathcal{L}}_s), \text{ i.e., low priority} \end{cases}$$

since all transmissions over $\bar{\mathcal{L}}_s$ were not successful, and all the other links scheduled at time-slot s (e.g., $\mathcal{L}_s' \cup (\tilde{\mathcal{L}}_s \setminus \bar{\mathcal{L}}_s)$) were successful. In particular, $r_{l's}[t+1] = 0$, since $l' \in \mathcal{L}_s'$.

Note that since $C^\star$ is feasible, the transmissions (with high priority) over $\bar{\mathcal{L}}_s$ will have successful RTS-H/CTS-H signaling at Stage 1. In particular, high priority RTS/CTS signaling over $\bar{\mathcal{L}}_s \setminus \bar{\mathcal{L}}_s^{l'}$ does not affect the transmission over link l', since none of the links in $\bar{\mathcal{L}}_s \setminus \bar{\mathcal{L}}_s^{l'}$ shares a contention group with l'. Thus, it is enough to see the transmission over $\bar{\mathcal{L}}_s^{l'}$ to investigate success or failure of transmission over l' at frame $t+1$. Then, from the time-slot release rule (Rule 6.4.2) in Section 6.4.2, the transmission over l' will be unsuccessful, and thus will be deferred. This completes the proof. □

Proof of Theorem 6.5.1

Proof. Consider an infeasible initial contention matrix, $C[0]$ (otherwise, the result immediately follows). Since $C[0]$ is infeasible and the offered load is feasible, we can find a feasible contention matrix, $C^\star$. We shall construct a finite sequence of times $0 = t_1 < t_2 < \cdots < t_n < \infty$, such that with positive probability $C[t_n]$ is feasible.

Then, it suffices to show the following: suppose $C[t_i]$ is not feasible. Then, we will find a time $t_{i+1},\ t_i < t_{i+1} < \infty$, such that with positive probability, either $D(C[t_{i+1}], C^\star) = D(C[t_i], C^\star) - 1$, or $C[t_{i+1}]$ is feasible (i.e., $C[t_{i+1}] = C^{\star\star} \neq C^\star$ for a feasible contention matrix $C^{\star\star}$). Note that since $D(C[t_1], C^\star)$ is finite and positive, this implies that after a finite time, the process eventually reach an absorbing state with positive probability.

Let $\mathcal{L}_u[t_i]$ denote the set of such unsatisfied links. Further, let

$$\mathcal{L}_g[t_i] \quad = \quad \{l \in \mathcal{L} \mid \vec{c}_{l\cdot}[t_i] = \vec{c}^{\star}_{l\cdot}\},$$

i.e., the set of links whose slot schedules (i.e., row vector of a contention matrix) are equal to those in $C^\star$. For notational simplicity, we henceforth omit the frame index, $[t_i]$, in the use of $\mathcal{L}_g[t_i]$ and $\mathcal{L}_u[t_i]$, unless explicitly needed.

Since $C[t_i]$ is not feasible, there must be at least an unsatisfied link, $l_u \in \mathcal{L}_u$. We choose $l_u \in \mathcal{L}_u \setminus \mathcal{L}_g$ if $\mathcal{L}_u \setminus \mathcal{L}_g \neq \emptyset$, and choose $l_u \in \mathcal{L}_g$ otherwise. Further, as l_u is not satisfied, there must exist a time-slot s_u such that a transmission over link l_u on slot s_u was not successful.
Case 1: $l_u \in \mathcal{L}_u \setminus \mathcal{L}_g$. In this case, we again consider two sub-cases based on whether $c^\star_{l_u s_u}$ = 0 or 1 (i.e., we consider whether the unsuccessful transmission over l_u on time-slot s_u is scheduled or not by $C^\star$).

(i) If $c^\star_{l_u s_u} = 0$, then since $l_u \notin \mathcal{L}_g$, there is a slot $s_m \neq s_u$, such that $c^\star_{l_u s_m} = 1$ and $c_{l_u s_m}[t_i] = 0$. Note that the slot schedules over l_u, $\vec{c}^{\star}_{l_u\cdot}$ and $\vec{c}_{l_u\cdot}[t_i]$ must have the same number of 1's. Now from Property 6.4.1, there is a positive probability that at frame $t_i + 1$ we have $C[t_i + 1]$, such that

$$c_{ls}[t_i + 1] \quad = \quad \begin{cases} 0 & \text{if } l = l_u,\ s = s_u\ , \\ 1 & \text{if } l = l_u,\ s = s_m\ , \\ c_{ls}[t_i] & \text{otherwise,} \end{cases}$$

i.e., the transmission originally scheduled over link l_u on slot s_u is moved to slot s_m at frame $t_i + 1$. Now, letting $t_{i+1} = t_i + 1$, we have $D(C[t_{i+1}], C^\star) = D(C[t_i], C^\star) - 1$.

(ii) $c^\star_{l_u s_u} = 1$. We first let $\mathcal{L}'_{s_u}$ denote the set of links, each of which has scheduled trans-

mission on time-slot s_u by $C[t_i]$, but not by $C^\star$, i.e.,

$$\mathcal{L}'_{s_u} \quad = \quad \{l \in \mathcal{L} \mid c_{ls_u} = 1,\ c^\star_{ls_u} = 0\}.$$

Then, again there are two cases in (ii): (a) there exists a link in $\mathcal{L}'_{s_u}$, which has the unsuccessful transmission on time-slot s_u, or (b) all the links in $\mathcal{L}'_{s_u}$ have successful transmissions at time-slot s_u.

(a): If there exists a link $l_F \in \mathcal{L}'_{s_u}$, over which the scheduled transmission on time-slot s_u was unsuccessful, then we can apply the similar argument to that in (i) (i.e., we move the unsuccessful transmission over l_F on s_u to other time-slot ($\neq s_u$), where a transmission is scheduled by $C^\star$) to have that $D(C[t_{i+1} = t_i + 1], C^\star) = D(C[t_i], C^\star) - 1$, since $\mathcal{L}'_{s_u} \cap \mathcal{L}_g = \emptyset$.

(b): If all the links in $\mathcal{L}'_{s_u}$ have successful transmissions on time-slot s_u, then from Lemma 6.5.1, (by letting $\bar{l} \equiv l_u$, and $\mathcal{L}'_s \equiv \mathcal{L}'_{s_u}$), there is a positive probability that there exists a link $l' \in \mathcal{L}'_{s_u}$, such that at the next frame $t_i + 1$, $C[t_i + 1] = C[t_i]$ (i.e., $D(C[t_i + 1], C^\star) = D(C[t_i], C^\star)$) and the transmission over l' is *unsuccessful.* Then, at the frame $t_i + 1$, $l' \notin \mathcal{L}_g[t_i + 1]$, $c^\star_{l's_u}[t_i + 1] = 0$ and l' is unsatisfied, which corresponds to **Case 1**(i). Thus, by letting $t_{i+1} = t_i + 2$, we have $D(C[t_{i+1}], C^\star) = D(C[t_i], C^\star) - 1$.

Case 2: $l_u \in \mathcal{L}_g \subset \mathcal{L}_u$. Note that the fact we are in this case implies that all the links in $\mathcal{L}_u \setminus \mathcal{L}_g$ are *satisfied*, since we always choose first an unsatisfied link in $\mathcal{L}_u \setminus \mathcal{L}_g$ by construction.

Then, using the same definition of $\mathcal{L}'_{s_u}$ as that in **Case 1**(ii), this case corresponds to **Case 1**(ii)(b), i.e., when all the links in $\mathcal{L}'_{s_u}$ have successful transmissions on time-slot s_u. Thus, again based on Lemma 6.5.1, with a positive probability we have $D(C[t_{i+1}], C^\star) = D(C[t_i], C^\star) - 1$, by letting $t_{i+1} = t_i + 2$. This completes the proof. □

Proof of Theorem 6.5.2

Proof. Let C be any initial contention matrix, and $C^\star$ be a feasible contention matrix. As in

Definition 6.5.1, we let

$$D(C, C^\star) \quad = \quad \sum_{l=1}^{|\mathcal{L}|} \rho_l - \sum_{l=1}^{|\mathcal{L}|} \sum_{s=1}^{F} c_{ls} \times c^\star_{ls}.$$

Let $n_\rho = \sum_{l=1}^{|\mathcal{L}|} \rho_l$ be the total number of loads.

Recall that the proof of Theorem 6.5.1 implies that for the current frame i, we have

$$\Pr\Big\{D(C[i+2], C^\star) = D(C[i], C^\star) - 1\Big\} \quad \geq \quad (1/F)^{2n_\rho},$$

i.e., over an interval of two frames, with at least probability $q = (1/F)^{2n_\rho}$, $D(C[i], C^\star)$ decreases by 1, since in the worst cast, each transmission was unsuccessful, and a time-slot is randomly chosen for each transmission with probability $1/F$, and we nave n_ρ number of scheduled transmissions, and this happens over two frames.

This also implies that with at least probability of $q^{D(C[0],C^\star)}$, the system will converge to $C^\star$ within $2 \times D(C[0], C^\star)$ frames. Note that convergence to a feasible schedule (possibly different from $C^\star$) could occur much earlier, since there could be multiple feasible contention matrices.

Further, note that for *any* contention matrix C, we have

$$D(C, C^\star) \geq n_\rho.$$

Thus, we have for any contention matrix C, we have $\forall C \in \mathcal{C}, \forall C^\star \in \mathcal{C}^\star$, (where $\mathcal{C}$ and $\mathcal{C}^\star$ is the set of all possible and feasible contention matrices, respectively)

$$\Pr\Big\{\tau(C) \leq 2n_\rho\Big\} \geq \Pr\Big\{\tau(C) \leq 2D(C, C^\star)\Big\} \geq q^{D(C,C^\star)} \geq q^{n_\rho}.$$

Now, let us study the evolution of the contention matrix at frames $\{0, 2n_\rho, 4n_\rho, \cdots\}$. Observe that at any frame $2t \cdot n_\rho$ it will converge to a feasible contention matrix within up to frame $2(t+1) \cdot 2n_\rho$ with probability at least $q^{n_\rho^2}$. This immediately provides a exponentially

fast upper bound on the convergence rate, i.e., $\forall C \in \mathcal{C}$,

$$\Pr\left\{\tau(C) > 2t \times n_\rho\right\} \quad \leq \quad (1 - q^{n_\rho^2})^t.$$

By letting $p = 1 - q^{n_\rho^2}$, and $K = 2n_\rho$, the result follows. □

Chapter 7

Concluding Remarks

Fluid models have been widely used, and shown to be efficient and accurate in the modeling, analysis, and design of the Internet. In literature, much of this work has focused on the design of end-host controllers and control algorithms at routers (marking functions) for the stable end-to-end operation over the Internet. However, there is a significant fraction of uncontrolled flows such as real-time video and audio flows in the Internet, and the effect of those uncontrolled flows on the design and simulation of the Internet cannot be ignored in the use of fluid models.

In Chapter 2, we have shown that the randomness due to short and unresponsive flows in the Internet is sufficient to decouple the dynamics of the router queues from those of the end controllers. This implies that a time-scale decomposition naturally occurs such that the dynamics of the router manifest only through their statistical steady-state behavior. This time-scale decomposition implies that a queue-length based marking function such as Random Early Detection (RED) or Random Exponential Marking (REM) have an equivalent form which depend only on the data arrival rate from the end-systems and do not depend on the queue dynamics.

Using the new rate-based fluid model in Chapter 2, in Chapter 3, we have presented a new approach to the design of hybrid packet/fluid simulation. Under a fast queue regime, our approach enables us to simulate large-scale systems, where the simulation step-size is governed only by the time-scale of the end-systems, and not that of the queueing dynamics at the intermediate routers. This gives us a significant reduction of both simulation time

complexity and implementation complexity for real-time emulation, compared with a queue tracking based hybrid simulation.

However, it is possible that we can be in a regime where fast queue regimes do not occur (e.g., by having large RED queue threshold parameters), where FluNet does not seem to perform well. In a real network, depending on the number of traversing flows and capacity at intermediate routers, fast and slow queue regimes can occur at different times and routers. In this case, queue tracking based fluid simulation and rate model based simulation should be chosen appropriately. A simple approach to differentiate between both regimes is to measure the number of regenerative cycles in a chosen step-size, and apply one of both fluid simulation models to mark/drop the packet at the intermediate routers.

Further, queueing delays could be significant with such a large queue operating size, which are ignored in the current implementation of FluNet (note, however, that FluNet can incorporate mean queueing delays, simply by adding an extra parameter to the round-trip propagation delay). In this case, a more accurate approach is to implement fluid queues to track the queue variation, and determine marking/dropping decisions based on the queue length (at the cost of having state in the simulator to keep track of the queue lengths). Thus, we believe that a good approach to hybrid simulation is to use both queue based method (such as in QFM [16]) as well as rate based method (such as FluNet), depending on the type of the system under study.

In Chapter 4, we have first quantified the trade-off between stability for controlled flows and QoS-guarantee for uncontrolled real-time flows as a function of marking elasticity. The results indicate that some marking functions may be "uniformly" better than others. In particular, among the marking functions that we have compared, our bounds indicate that a rate based version of REM seems to provide the largest local-stability region for *any* given QoS requirement. Next, we have compared the capacity required at a router with only FIFO scheduling versus a router with priority scheduling (priority given to the real-time flows) for supporting a given QoS requirement. We have quantified the "scheduling-gain" of priority scheduling over FIFO scheduling, as a function of marking elasticity. We have shown that this scheduling gain decreases with more elastic marking

functions.

While much of the research based on fluid models has focused on wire-line networks, a growing area of research is that of wireless multi-hop networks.

In Chapter 5, we have first investigated fluid models for MAC and appropriate congestion control models for multi-hop wireless networks. Based on an optimization framework with constraints that arise from the multi-hop wireless network, we have proposed a hop-by-hop congestion control algorithm, and have studied its properties on the stability and peak buffer requirement. In the absence of delay, we have shown that this algorithm are globally stable using a Lyapunov function based approach. Next, in the presence of delay, we have shown that the hop-by-hop control algorithm has the property of spatial spreading. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. We derive bounds on the "peak load" at a node, both with hop-by-hop control, as well as with end-to-end control, and have shown that significant gains are to be had with the hop-by-hop scheme

In Chapter 5, we had assumed that a fluid model at the MAC layer, and studied the properties of the congestion control algorithm. In Chapter 6, we have considered the complementary discrete problem of determining dynamic schedulers, such that over a long time-scale it results in a fluid model. In this context, we have addressed the problem of dynamic scheduling for multi-rate TDMA wireless networks with arbitrary topologies. We have developed a synchronous contention-based MAC algorithm (DCAMA) that provably converges to a conflict-free schedule for any feasible load and for arbitrary topologies. The key mechanisms that enables us to achieve this are a synchronous two-level priority RTS/CTS based contention scheme and randomized selection of time-slots. Based on this algorithm, we have proposed heuristics (ADCAMA, which also provably converges) that improve the convergence time by biasing time-slot access probabilities based on past contention history.

An issue that we do not directly address in this thesis is the behavior of these algorithms when the load is not feasible. To handle such cases, we will need to combine admission control strategies (long time-scale control) along with the MAC algorithms (short time-scale resource allocation) to ensure that a feasible solution exists. It would be of interest in the

future to study admission control policies in conjunction with the (A)DCAMA algorithm.

Bibliography

[1] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[2] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000, vol. 3, pp. 1323–1332.

[3] G. Vinnicombe, "On the stability of end-to-end congestion control for the Internet," 2001, University of Cambridge Technical Report.

[4] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.

[5] L. Massoulie, "Stability of distributed congestion control with heterogenous feedback delays," *Technical Report, Microsoft Research, Cambridge, UK*, 2000.

[6] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *Proceedings of IEEE Conference on Decision and Control*, December 2001, vol. 1, pp. 185–190.

[7] R. Johari and D. Tan, "End-to-end congestion control for the Internet: Delays and stability," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, December 2001.

[8] S. Shakkottai, R. Srikant, and S. Meyn, "Bounds on the throughput of congestion

controllers in the presence of feedback delay," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, 2003.

[9] S. Shakkottai and R. Srikant, "Mean FDE models for Internet congestion control under a many-flows regime," *IEEE Transactions on Information Theory*, vol. 50, no. 6, June 2004.

[10] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, pp. 861–875, December 1999.

[11] Steven H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[12] V. Misra, W.-B. Gong, and D.Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of ACM SIGCOMM*, 2000.

[13] D. M. Nicol and G. Yan, "Discrete event fluid modeling of background TCP traffic," *ACM Transactions on Modeling and Computer Simulation*, vol. 14, no. 3, July 2004.

[14] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid models and solutions for large-scale ip networks," in *Proceedings of ACM SIGMETRICS*, June 2003.

[15] F. Baccelli and D. Hong, "Flow level simulation of large ip networks," in *Proceedings of INFOCOM*, San Francisco, CA, April 2003.

[16] Y. Gu, Y. Liu, and D. Towsley, "On Integrating Fluid Models with Packet Simulation," in *Proceedings of IEEE INFOCOM*, March 2004.

[17] T. Yung, J. Martin, M. Takai, and R. Bagrodia, "Integration of fluidbased analytical model with packet-level simulation for analysis of computer networks," in *Proceedings of SPIE*, 2001.

[18] B. Melamed, S. Pan, and Y. Wardi, "Hybrid discrete-continuous fluid-flow simulation," in *Proceedings of ITCOM, Scalability and Traffic Control in IP Networks*, August 2001.

[19] G. Riley, R. Fujimoto, M. Ammar, K. Permula, and D. Xu, "Distributed network simulations using the dynamic simulation backplane," in *Proceedings of International Conference of Distributed Computing Systems*, 2001.

[20] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems modeling framework for fast and accurate simulation of data communication networks," in *Proceedings of ACM SIGMETRICS*, 2003.

[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[22] F.P. Kelly, "Mathematical modelling of the Internet," in *Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid)*, Berlin, 2001, pp. 685–702, Springer-Verlag.

[23] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, pp. 28–43, February 2002.

[24] S. Deb, S. Shakkottai, and R. Srikant, "Stability and convergence of TCP-like congestion controllers in a many-flows," in *Proceedings of INFOCOM*, San Francisco, CA, 2003.

[25] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," in *Proceedings of IEEE INFOCOM*, New York, NY, 2002, vol. 1, pp. 239–248.

[26] S. Shakkottai and R. Srikant, "How good are deterministic fluid models of Internet congestion control?," in *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[27] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001, pp. 123–134.

[28] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, May/June 2001.

[29] P. Tinnakornsrisuphap and A. Makowski, "Limit behavior of ECN/RED gateways under a large number of TCP flows," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, 2003.

[30] J. Cao and K. Ramanan, "A poisson limit for buffer overflow probabilities," in *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[31] M. Mandjes and J. H. Kim, "Large deviations for small buffers: an insensitivity result," *Queueing Systems*, vol. 37, pp. 349–362, 2001.

[32] S. Deb and R. Srikant, "Rate-based versus Queue-based models of congestion control," in *Proceedings of ACM SIGMETRICS*, 2004.

[33] "Ns-2," http://www.isi.edu/nsnam/ns/.

[34] PDNS, "Parallel and Distributed NS," http://www.cc.gatech.edu/computing/compass/pdns/.

[35] GloMoSim, ," http://pcl.cs.ucla.edu/projects/glomosim/.

[36] QualNet, ," http://www.scalable-networks.com.

[37] D.J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*, Springer-Verlag, New York, NY, 1988.

[38] P. Billingsley, *Convergence of Probability Measures*, Wiley-Interscience, 1999.

[39] W. L. Smith, "Regenerative stochastic processes," in *Proceedings of The Royal Society of London*, 1955, vol. A 232, pp. 6–31.

[40] S. I. Resnick, *Adventures in Stochastic Processes*, Birkhauser, Boston, 1992.

[41] R. W. Wolff, *Stochastic Modeling and the Theory of Queues*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[42] U. Mocci J. Roberts and J. Virtamo, *Broadband Network Teletraffic, Final Report of Action COST 242*, Birkhauser, Boston, 1992.

[43] J. Virtamo, "Numerical evaluation of the distribution of unfinished work in an M/D/1 system," *Electronics Letters*, vol. 31, no. 7, pp. 531–532, 1995.

[44] S. Floyd, "Thoughts on the evolution of tcp in the internet," Invited talk at the Second International Workshop on Protocols for Fast Long-Distance Networks, 2004.

[45] C. V. Hollot, Y. Liu, V. Misra, and D. Towsley, "Unresponsive flows and AQM performance," in *Proceedings of INFOCOM*, San Francisco, CA, April 2003, vol. 1, pp. 85–95.

[46] H. Kim and J. C. Hou, "Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation," in *Proceedings of IEEE INFOCOM*, March 2004.

[47] Network Simulators, ," http://www.icir.org/models/simulators.html.

[48] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–23, October 1994.

[49] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik, "Shrink: A method for scaleable performance prediction and efficient network simulation," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, 2003.

[50] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proceedings of ACM SIGCOMM*, 2004.

[51] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part iii: Routers with very small buffers," 2005.

[52] CAIDA, ," http://www.caida.org.

[53] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," in *Proceeding of INFOCOM*, March 2004.

[54] Y. Yi, S. Deb, and S. Shakkottai, "Time-scale decomposition and rate-based marking," 2005, To appear at IEEE/ACM Transactions on Networking.

[55] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," in *Proceedings of Next Generation Internet Networks*, 2005.

[56] H. Jiang and C. Dovrolis, "The origin of TCP traffic burstiness in short time scales," Tech. Rep., 2004, Available at "http://www.cercs.gatech.edu/tech-reports/tr2004/git-cercs-04-09.pdf".

[57] C. Villamizar and C. Song, "High performance tcp in ansnet," *ACM SIGCOMM Computer Communications Review*, vol. 24, no. 5, pp. 45–60, 1994.

[58] S. Floyd and E. Kohler, "Internet research needs better models," in *HotNets-I*, October 2002.

[59] P. Wood, "A libpcap version whith supports mmap mode," http://public.lanl.gov/cpw/.

[60] Iperf, ," http://dast.nlanr.net/Projects/Iperf/.

[61] L. Ying, G. Dullerud, and R. Srikant, "Global stability of internet congestion controllers with heterogeneous delays," in *Proceedings of American Contrl Conference*, June 2004.

[62] C. V. Hollot and Y. Chait, "Nonlinear stability analysis for a class of TCP/AQM schemes," in *Proceedings of the IEEE Conference on Decision and Control*, December 2001.

[63] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," December 1998, RFC 2475.

[64] L. Massouli and J. Roberts, "Bandwidth sharing: objectives and algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 320–328, 2002.

[65] G. de Veciana and J. Walrand, "Effective bandwidths: Call admission, traffic policing and filtering for ATM networks," *Queueing Systems Theory and Applications*, vol. 20, pp. 37–59, 1995.

[66] D. D. Botvich and N. G. Duffield, "Large deviations, economies of scale, and the shape of the loss curve in large multiplexers," *Queueing Systems*, vol. 20, pp. 293–320, 1995.

[67] C. Courcoubetis and R. Weber, "Buffer overflow asymtotics for a switch handling many traffic sources," *Journal of Applied Probability*, vol. 33, pp. 886–903, 1996.

[68] G. Kesidis, J. Walrand, and C-S. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 424–428, 1993.

[69] N. Likhanov and R. Mazumdar, "Cell loss asymptotics for buffers fed with a large number of independent stationary sources," *Journal of Applied Probability*, vol. 36, pp. 86–96, 1999.

[70] A. Shwartz and A. Weiss, *Large Deviations for Performance Analysis*, Chapman and Hall, New York, NY, 1995.

[71] D. Wischik, "Sample path large deviations for queues with many inputs," *Annals of Applied Probability*, 2000.

[72] Y. Yi, S. Deb, and S. Shakkottai, "Short queue behavior and rate based marking," in *Proceedings of the 38th Conference on Information Sciences and Systems*, March 2004.

[73] A. J. Ganesh and N. O'connell, "A large deviation principle with queueing applications," 1997, Technical Report HPL-BRIMS-9705, BRIMS, Hewlett Packard Labs, Bristol.

[74] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proceedings of IEEE/ACM Mobicom*, August 1999, pp. 219–230.

[75] P. Bender, P. Black, M. Grob, R. Padovani, N Sindhushayana, and A. Viterbi, "CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users," *IEEE Communications Magazine*, pp. 70–77, July 2000.

[76] IEEE Standard 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1997.

[77] P. P. Mishra and H. Kanakia, "A hop by hop rate based congestion control scheme," in *Proceedings of ACM SIGCOMM*, August 1992.

[78] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation and statistical multiplexing," in *Proceeding of ACM SIGCOMM*, 1994, pp. 101–114.

[79] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 236–250, February 1995.

[80] S. Sarkar and L. Tassiulas, "Back pressure based multicast scheduling for fair bandwidth allocation," in *Proceedings of IEEE INFOCOM*, 2001.

[81] S. Borst, "User-level performance of channel-aware scheduling algorithms in wireless data networks," in *In Proceedings of INFOCOM*, 2003.

[82] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, pp. 466–478, March 1993.

[83] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "On the scalability of fair queueing," in *ACM HotNets-III*, San Diego, November 2004.

[84] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks," in *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[85] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proceedings of IEEE Conference on Decision and Control*, Bahamas, 2004.

[86] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media acess control design for ad hoc wireless networks," in *Proceeding of IEEE INFOCOM*, Miami, FL, 2005.

[87] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Upper Saddle River, NJ, 2002.

[88] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem," in *Proceeding of ACM MobiCom*, 2003.

[89] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, 1988.

[90] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[91] S. Kunniyur and R. Srikant, "A time-scale decomposition approach to adaptive ECN marking," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 882–894, June 2002.

[92] G. V. Zaruba, S. Basagni, and I. Chlamtac, "Bluetrees - scatternet formation to enable bluetooth-based ad hoc networks," in *Proceedings of ICC*, 2001.

[93] R. Guerin, J. Rank, S. Sarkar, and E. Vergetis, "Forming connected topologies in bluetooth adhoc networks," in *Proceedings of ITC 18*, 2003.

[94] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proceedings of INFOCOM*, 2000.

[95] V. S. Raghavan S. Kumar and J. Deng, "Medium access control protocols for ad-hoc wireless networks: A survey," *Elsevier Ad-Hoc Networks Journal*, 2005, To appear.

[96] T. J. Shepard, "A channel access scheme for large dense packet radio networks," in *Prooceeding of SIGCOMM*, 1996.

[97] R. Rozovsky and P. R. Kumar, "Seedex: a mac protocol for ad hoc networks," in *Proceeding of MobiHoc*, 2001.

[98] X. Yang and G. de Veciana, "Inducing spatial clustering in mac contention for spread spectrum ad hoc networks," in *Proceedings of MobiHoc*, 2005.

[99] J. Stine and G. de Veciana, "A paradigm for quality-of-service in wireless ad hoc networks using synchronous signaling and node states," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1301–1321, September 2004.

[100] C. Zhu and M. S. Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," *Wireless Networks*, vol. 7, no. 4, pp. 371–384, 2001.

[101] Z. Tang and J. J. Garcia-Luna-Aceves, "A protocol for topology-dependent transmission scheduling in wireless networks," in *Proceedings of WCNC*, 1999.

[102] L. Bao and J. J. Garcia-Luna-Aceves, "Distributed dynamic channel access scheduling for ad hoc networks," *Journal of Parallel Distributed Computing*, vol. 63, no. 1, pp. 3–14, 2003.

[103] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *Proceedings of MOBIHOC*, 2005.

[104] P. While, "RSVP and integrated services in the internet: A tutorial," *IEEE Communications Magazine*, May 1997.

[105] I. Chlamtac and A. Lerner, "Link allocation in mobile radio networks with noisy channel," in *Proceedings of INFOCOM*, April 1986.

[106] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proceedings of IEEE INFOCOM*, 2005.

[107] A. Panconesi and A. Srinivasan, "Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds," *SIAM Journal of Computing*, vol. 26, no. 2, 1997.

[108] D. A. Grable and A. Panconesi, "Nearly optimal distributed edge colouring in o(log log n) rounds," in *Proceedings of ACM-SIAM symposium on Discrete algorithms*, 1997.

[109] S. Ramanathan, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, no. 2, pp. 81–94, 1999.

[110] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1694–1701, 1981.

[111] A. Ephremides and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Transactions on Communications*, 1990.

[112] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166–177, 1993.

[113] R. Ramaswami and K.K. Parhi, "Distributed scheduling of broadcasts in a radio network," in *Proceeding of INFOCOM*, 1989.

[114] S. O. Krumke, M. V. Marathe, and S. S. Ravi, "Models and approximation algorithms for channel assignment in radio networks," *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

[115] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet radio networks," *IEEE Transactions on Computers*, vol. 38, no. 10, 1989.

[116] I. Chlamtac and S. Kutten, "A spatial reuse TDMA/FDMA for mobile multihop radio nertworks," in *Proceedings of INFOCOM*, March 1985.

[117] U. Feige, E. Ofek, and U. Wieder, "approximating maximum edge coloring in multigraphs," *In APPROX, volume 2462 of LNCS*, pp. 108–121, 2002.

[118] P. Sanders and D. Steurer, "An asymptotic approximation scheme for multigraph edge coloring," in *Proceedings of ACM-SIAM symposium on Discrete algorithms*, 2005.

[119] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *Proceedings of ACM SIGCOMM*, September 1994.

[120] J. H. Ju and V. O. K. Li, "An optimal topology-transparent scheduling method in multihop packet radio," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, 1998.

[121] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, 1994.

[122] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media sccess for multirate ad hoc networks," in *Proceedings of ACM MobiCom*, 2002.

Vita

Yung Yi joined the Department of Electrical and Computer Engineering at the University of Texas at Austin in the fall of 2002 as a Ph.D. student working with Dr. Sanjay Shakkottai. He received the B.S.E, and the M.S.E in School of Computer Science and Engineering from Seoul National University, South Korea in 1997, and 1999, respectively.

While at Seoul National University, he was part of various research projects on computer networking such as QoS multicast routing, multimedia broadcasting, streaming and flow control protocol, packet scheduling, and next generation Internet. He was also a visiting researcher at the Department of Computer Science, North Carolina State University, in 1999.

In addition to academic areas, he worked as a programmer at Togabi-Korea Inc and as a network security researcher at *Internet Crime Investigation Center of Public at the Supreme Public Prosecutor's Office* in South Korea.

His current research interests include ad-hoc and sensor networks, scheduling and QoS in wireless networks, congestion control in the Internet, and resource allocation for heterogeneous networks.

Permanent Address: 12113 Metric Blvd APT 527, Texas, Austin, 79758

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department

of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.